ioProgrammo

MODEM A RADIOFREQUENZA
COSTRUIAMOLO E REALIZZIAMO UNA RETE WIRELESS

VERSIONE PLUS ✔ RIVISTA+LIBRO+CD €14,90 — VERSIONE STANDARD ☐ RIVISTA+CD €6,90

Poste Italiane • Spedizione in a.p. - 45% • art. 2 comma 20/b legge 662/96 - AUT. N. DCDC/033/01/CS/CAL Periodicità mensile • DICEMBRE 2004 • ANNO VIII, N.11 (86)

CODICE WEBCAM

DALLA 'A' ALLA 'Z' TUTTO QUELLO CHE SERVE PER PROGRAMMARE UNA VIDEOCAMERA

- ✔ I primi passi per acquisire il video
- ✔ Salviamo foto e filmati sull'hard-disk
- ✔ Usiamo una libreria gratuita per il riconoscimento facciale

SORGENTI Visual Basic nel CD

CREA DA ZERO UN GIOCO 3D

Le prime 20 righe di codice per diventare programmatore di videogiochi

ISSN 1128-594X
ioProgrammo Anno VIII - N° 11 (86) • € 14,90

■ MOBILE

Il meteo nel cellulare

Le previsioni "piovono" da Internet!

■ ECLIPSE

JAVA e le interfacce

Introduzione alle Standard Widget Toolkit

WEB SERVICES

- Un convertitore di valuta in tempo reale
- Calcolare il codice fiscale con una sola istruzione

AL VIA IL CORSO

JAVA

SVILUPPARE UN EDITOR FOTOGRAFICO

Introduzione a JAI, la libreria di Sun per il fotoritocco

LA GESTIONE DEGLI ERRORI IMPREVEDIBILI

Come costruire applicazioni a prova di utenti maldestri!

.NET

VISUAL BASIC GRAFICI STATISTICI

Usiamo MS Chart per produrre diagrammi in stile Excel

REALIZZARE UN MESSENGER IN C#

Impariamo come inviare e ricevere messaggi con file in allegato

EXPRESS & TRUCCHI

oltre 10 pagine di soluzioni da consumare al volo

REPORTAGE SUN GLASS: TUTTE LE NOVITÀ DEL NUOVO SISTEMA GRAFICO

Anno VIII - n. 11 (86) Dicembre 2004

▼ Web Services: grande illusione?

Dovevano essere la nuova terra promessa, il nuovo Eldorado della programmazione. Dovevano diffondersi a macchia d'olio e trasformare radicalmente il concetto stesso di programmazione. Sarebbero dovuti arrivare migliaia e migliaia di servizi, gratuiti e a pagamento, che avrebbero risolto nuove problematiche e aiutato gli sviluppatori nei compiti più consueti.
Niente di tutto questo è (ancora) accaduto. Come tutte le promesse troppo grandi, i salti tecnologici troppo accentuati, i Web Service sono rimasti dominio di pochi e non sono entrati nella programmazione "di massa". All'inizio, WS era quasi sinonimo di XML e, come quest'ultimo, i servizi Web parevano essere la panacea per tutti i problemi di comunicazione e integrazione: dalla produttività personale alla grande azienda, sembrava che tutti avrebbero avuto benefici da queste innovazioni. Non è stato così. Utilissimi in ambiti definiti come l'integrazione fra sistemi eterogenei o distribuiti su base geografica, i WS hanno mostrato la corda per problemi di integrazione su scala più ridotta, in cui tecnologie proprietarie e meno eleganti, ma più semplici, continuano a farla da padrone. Tutto questo, senza nulla togliere alla validità di SOAP e delle altre tecnologie legate ai Web Services: la direzione è probabilmente quella giusta, sono da rivedere le previsioni su tempi e dimensioni del fenomeno.

Raffaele del Monaco

CD WEB
nome_file.zip
cdrom.ioprogrammo.it

All'inizio di ogni articolo, troverete un nuovo simbolo che indicherà la presenza di codice e/o software allegato, che saranno presenti sia sul CD (nella posizione di sempre *\soft\codice* e *\soft\tools*) sia sul Web, all'indirizzo http://cdrom.ioprogrammo.it.
Per scaricare software e codice da Internet, ogni mese indicheremo una password differente. Per il numero che avete fra le mani la combinazione è:

Username: **C3PO** *Password:* **D3BO**

ioProgrammo

Anno VIII - N.ro 11 (86) - Dicembre 2004 - Periodicità: Mensile
Reg. Trib. di CS al n.ro 593 del 11 Febbraio 1997
Cod. ISSN 1128-594X
E-mail: ioprogrammo@edmaster.it
http://www.edmaster.it/ioprogrammo
http://www.ioprogrammo.it

Direttore Editoriale Massimo Sesti
Direttore Responsabile Massimo Sesti
MarCom Director Luca Perfetti
Marketing Manager Antonio Meduri
Responsabile Editoriale Gianmarco Bruni
Editor Gianfranco Forlino
Coordinamento redazionale Raffaele del Monaco
Redazione Fabio Farnesi
Collaboratori M. Autiero, A. Baldini, M. Bigatti, L. Buono, C. De Sio Cesari, F. Diotallevi, F. Grimaldi, F. Lippo, M. Locuratolo, A. Marroccelli, L. Mattei, F. Mestrone, G. Naccarato, F. Paparoni, P. Perrotta, L. Spuntoni, E. Tacchini, F. Vaccaro, C. F. Zoffoli.
Segreteria di Redazione Veronica Longo

Realizzazione grafica Cromatika S.r.l.
Responsabile grafico: Paolo Cristiano
Coordinamento tecnico: Giancarlo Sicilia
Illustrazioni: M. Veltri
Impaginazione elettronica: Aurelio Monaco

"Rispettare l'uomo e l'ambiente in cui esso vive e lavora è una parte di tutto ciò che facciamo e di ogni decisione che prendiamo per assicurare che le nostre operazioni siano basate sul continuo miglioramento delle performance ambientali e sulla prevenzione dell'inquinamento"

Certificato UNI EN ISO 14001
N. 9191 CRMT

Realizzazione Multimediale SET S.r.l.
Coordinamento Tecnico Piero Mannelli
Realizzazione CD-Rom Paolo Iacona

Pubblicità Master Advertising s.r.l.
Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121207
e-mail advertising@edmaster.it
Sales Director: Max Scortegagna
Segreteria Ufficio Vendite Daisy Zonato

Editore Edizioni Master S.p.A.
Sede di Milano: Via Ariberto, 24 - 20123 Milano
Tel. 02 831212 - Fax 02 83121206
Sede di Rende: C.da Lecco, zona industriale - 87036 Rende (CS)
Amministratore Unico: Massimo Sesti

Abbonamento e arretrati
ITALIA: Abbonamento Annuale: ioProgrammo Basic (11 numeri) €49,90 sconto 35% sul prezzo di copertina di €75,90
ioProgrammo con Libro (11 numeri + libro) €79,90 sconto 40% sul prezzo di copertina di €133,90
Offerte valide fino al 31/01/05

ESTERO: Abbonamento Annuale: ioProgrammo Basic (11 numeri): € 151,80. ioProgrammo Plus (11 numeri + 11 libri): € 257,00
Costo arretrati (a copia): il doppio del prezzo di copertina + € 5,32 spese (spedizione con corriere). Prima di inviare i pagamenti, verificare la disponibilità delle copie arretrate allo 02 831212.
La richiesta contenente i Vs. dati anagrafici e il nome della rivista, dovrà essere inviata via fax allo 02 83121206, oppure via posta a EDIZIONI MASTER via Ariberto, 24 - 20123 Milano, dopo avere effettuato il pagamento, secondo le modalità di seguito elencate:

- **cc/p n.16821878** o **vaglia postale** (inviando copia della ricevuta del versamento insieme alla richiesta);
- **assegno bancario non trasferibile** (da inviarsi in busta chiusa insieme alla richiesta);
- **carta di credito**, circuito VISA, CARTASI', MASTERCARD/EUROCARD, (inviando la Vs. autorizzazione, il numero della carta, la data di scadenza e la Vs. sottoscrizione insieme alla richiesta).
- **bonifico bancario** intestato a Edizioni Master S.p.A. c/o Banca Credem S.p.a. c/c 01 000 000 5000 ABI 03032 CAB 80880 CIN Q (inviando copia della distinta insieme alla richiesta).

SI PREGA DI UTILIZZARE IL MODULO RICHIESTA ABBONAMENTO POSTO NELLE PAGINE INTERNE DELLA RIVISTA. L'abbonamento verrà attivato sul primo numero utile, successivo alla data della richiesta.
Sostituzioni: Inviare il CD-Rom difettoso in busta chiusa a:
Edizioni Master Servizio Clienti - Via Ariberto, 24 - 20123 Milano

Assistenza tecnica: ioprogrammo@edmaster.it

Servizio Abbonati:
☎ tel.02 831212
@ e-mail: servizioabbonati@edmaster.it

Stampa: Rotoeffe Via Variante di Cancelliera, 2/6 - Ariccia (Roma)
Stampa CD-Rom: Neotek S.r.l.- C.da Imperatore- zona ASI - Bisignano (CS)
Distributore esclusivo per l'Italia: Parrini & C S.p.A.
Via Vitorchiano, 81 - Roma

Finito di stampare nel mese di Novembre 2004

Edizioni Master edita: Idea Web, Go!OnLine Internet Magazine, Win Magazine, PC Fun extreme, Quale Computer, DVD Magazine, Office Magazine, La mia Barca, ioProgrammo, Linux Magazine, Softline Software World, HC Guida all'Home Cinema, MPC, Discovery DVD, Computer Games Gold, inDVD, I Fantastici CD-Rom, PC VideoGuide, I Corsi di Win Magazine, I Filmissimi in DVD, La mia videoteca, TV e Satellite, Win Extra, Home entertainment, Digital Japan Magazine, Digital Music, Horror Mania, ioProgrammo Extra, Le Collection.

News

GLI HACKER AGGIRANO GLI ANTIVIRUS

Tutte le software house più importanti stanno già correndo ai ripari. iDefense ha infatti appena annunciato di avere scoperto alcune falle nella tecnica di gestione dei file compressi. Sfruttando questo tipo di vulnerabilità, i programmatori di virus stanno già riprogettando le loro creature in modo che possano essere inseriti in file compressi manipolati e aggirare ogni tipo di controllo da parte degli antivirus.

RFID NEI PASSAPORTI AMERICANI

Il Dipartimento di Stato americano sta pianificando di installare un chip RFID all'interno dei passaporti. La sperimentazione dovrebbe partire in Gennaio e coinvolgere, almeno inizialmente, il corpo diplomatico americano.
RFID è l'acronimo di *Radio Frequency Identification*. Si tratta di un piccolissimo trasmettitore alimentato da un lettore specifico attraverso la sua antenna. Il codice trasmesso è la parte più innovativa del sistema, trattandosi di un numero identificativo univoco su tutta la gerarchia mondiale.
L'annuncio della volontà del Dipartimento di Stato americano di volere utilizzare RFID ha già provocato non poche polemiche, trattandosi di un argomento che coinvolge la privacy.
I detrattori di questa idea sostengono che chiunque dotato del lettore giusto potrebbe "downloadare" dal chip le informazioni personali che contiene.

GOOGLE RIPARA LA SUA DESKTOP SEARCH BAR

La nuova nata di casa Google, consente di indicizzare i contenuti di un hard disk locale, per poi sfruttare la stessa tecnica di ricerca presente sul Web sui propri documenti privati. Si tratta, di un'innovazione piuttosto interessante. Consente ricerche rapidissime su tutto il contenuto del proprio Hard Disk. Una comodità che già in molti stanno utilizzando per il proprio lavoro. Peccato che sia stata appena scoperta una falla piuttosto preoccupante. La Desktop Search Bar non impedirebbe l'esecuzione di codice pericoloso. Tanto che alcuni utenti si sono trovati al posto della classica pagina di ricerca di Google una pagine che gli chiedeva i propri dati personali e il proprio numero di carta di credito. Google è prontamente corsa ai ripari.
Al momento il bug sembra essere stato risolto.

GRAPHENE PER I CHIP DEL FUTURO

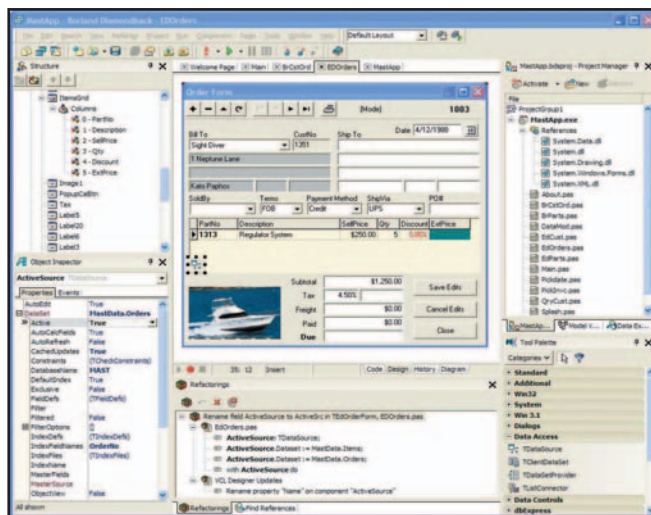
L'università di Manchester ha annunciato di avere scoperto una tecnica in grado di estrarre da un cristallo di grafite un singolo piano di atomi di carbonio. Il che darebbe luogo alla struttura più sottile al mondo: il Graphene.
L'interesse dei ricercatori appare alto.
Le proprietà conduttive del carbonio lo rendono infatti il materiale principe utilizzato nella realizzazione di chip e microchip.

SERVICE PACK 2 SOTTO ATTACCO

Sono due i nuovi buchi scoperti in Internet Explorer e che non consentirebbero nemmeno agli utenti che hanno installato il Service Pack 2 di rimanere tranquilli. Il primo riguarda un mancato controllo di validazione sulla procedura di Drag and Drop da Internet verso il proprio PC. Questo tipo di falla consentirebbe ad utenti maliziosi di inserire nelle proprie pagine Web degli script che potrebbero poi essere eseguiti anche in locale. Con la logica conseguenza che un utente che abbia scaricato le pagine infette sul proprio computer potrebbe trovarsi ad eseguire in locale, senza saperlo, del codice altamente pericolo-

DELPHI 2005 PRONTO AL LANCIO

Delphi 2005 introduce numerose novità. La più interessante delle quali e' probabilmente il supporto per C#. La flessibilita' del nuovo ambiente sara' duplice. Dal punto di vista del linguaggio, si potrà utilizzare indifferentemente C# o Delphi e, dal punto di vista della piattaforma, si potrà scegliere se compilare per Win32 o per .NET. Le innovazioni riguardano anche l'impostazione stessa dell'ambiente che, grazie all'integrazione di tecnologie come ALM (Application Lifecycle Management), mira a diventare uno strumento che copre l'intero ciclo di sviluppo. Significativi i miglioramenti ance negli strumenti di collaborazione e nelle funzioni di test delle unit. I sistemi di aiuto alla programmazione vanno da un help integrato pronto a soccorre lo sviluppatore durante la digitazione del codice, ad un piu' avanzato agente per la rilevazione degli errori. Delphi 2005 si offre come un'ottima soluzione per la realizzazione di interfacce verso ADO.NET. Gli sviluppatori più esperti potranno beneficiare di ECO II: una soluzione per .NET, basata sulla *Model Driver Architecture (MDA)*, che mira a introdurre un approccio RAD nelle applicazioni di livello enterprise.

www.borland.it

so per la propria macchina. La seconda falla riguarda, invece, un errore sulle impostazioni di sicurezza di Explorer. In particolare navigando sul Web e incorrendo in un particolare, tipo di link, ad esempio un file tipo *.hhk* costruito in modo adeguato, si rischia che la pagina in questione mandi in esecuzione un file html sul computer locale. I due bug, combinati, sono evidentemente pericolosissimi. Il primo consente di non validare alcuni script scaricati da Internet e potenzialmente dannosi; il secondo consente di eseguire, in locale, le pagine dannose che potreste avere, involontariamente scaricato. Al momento in cui scriviamo non esistono patch per queste falle.
Speriamo in un prossimo futuro.

http://secunia.com/advisories/12889/

WEB SERVICES: IL PRIMO PLUG-IN PER ECLIPSE

Systinet si avvia a rilasciare Systinet Developer for Eclipse 5.0 che sarà il primo plug-in per Eclipse orientato allo sviluppo di Web Service. Il tool seguirà l'intero ciclo di sviluppo-debug-runtime per applicazioni basate su Web Services. Tra le principali funzionalità del tool si segnalano: la generazione automatica dei client a partire da documenti WSDL; debugging sia server-side sia client-side; distribuzione dei file attraverso archivi standard Java; generazione dei documenti UDDI per la pubblicazione e la registrazione dei servizi.

www.systinet.com

UN'ETICHETTA PER I TELEFONI NOKIA

I tecnici del gigante finlandese sono impegnati in un progetto per l'inserimento di tag RFID nei cellulari. Nei tag saranno indicate informazioni legate al prodotto e potranno essere utilizzati per rintracciare la posizione del dispositivo. Questa mossa porterebbe i benefici dell'RFID anche in settori diversi dalla catena di distribuzione, cui fino ad ora è stata confinata: servizio alla clientela e gestione del marketing potrebbero trarne grandi vantaggi. Un prototipo è stato presentato al CTIA Wireless IT & Entertainment: basato su di un Nokia 5140, integrava un tag RFID le cui informazioni erano correlate ad un database centralizzato gestito da VeriSign. Il prototipo consentiva funzioni di inventario ed l'invio di informazioni personalizzate per il singolo dispositivo.

www.nokia.com

IBM ENTRA NELLA LIBERTY ALLIANCE

Fondata nel 2001 da Sun Microsystems, Liberty Alliance è un gruppo di grandi aziende, non solo informatiche, impegnato nello sviluppo di standard industriali per la verifica dell'identità degli utenti collegati via Web. IBM entra subito nel consiglio direttivo del gruppo, con l'intento di trovare un terreno comune fra gli standard fin qui sviluppati dalla Liberty Alliance e le specifiche adottate finora da IBM. Allo sforzo di integrazione di IBM, non corrisponde un eguale impegno di Microsoft che resta arroccata nella sua tecnologia di autenticazione, Passport, utilizzata dalla Microsoft stessa e dai suoi più vicini partner tecnologici.

www.ibm.it

SUN LANCIA IL PROGETTO EPSILON

Secondo Sun il mondo delle periferiche di tipo Wireless è una giungla *selvaggia*, dove ogni periferica è governata da software sviluppati con linguaggi diversi e il più delle volte antiquati.
Perciò il mondo Wireless è *selvaggio*, le comunicazioni poco protette e spesso instabili.
Perciò Sun ha messo in piedi il progetto Epsilon costituito da persone che in precedenza avevano già lavorato allo sviluppo di CDC – *Connected Device Configuration* – ossia una Java Virtual Machine ottimizzata per essere usata su periferiche con un basso livello di risorse. Il progetto Epsilon si propone di ottimizzare questa JVM anche per le periferiche di tipo Wireless. Poiché Java è un linguaggio conosciuto da un gran numero di sviluppatori, gia largamente diffuso su cellulari, PDA e palmari, una sua estensione al mondo delle periferiche wireless garantirebbe uno sviluppo omogeneo delle reti di questo tipo.
Ovviamente si tratta di aggiungere il supporto per interagire con sensori, periferiche e comunicazioni a onde radio continuando ad ottimizzare il codice affinché possa essere usato su sistemi con un limitato numero di risorse.

FILEMAKER SI AGGIORNA

FileMaker Inc. ha annunciato il rilascio degli update gratuiti per FileMaker Pro 7, FileMaker Developer 7, FileMaker Server 7 e di FileMaker Server 7 Advanced.
Le innovazioni più interessanti hanno riguardato la versione Pro, che vede oltre trecento innovazioni in pressoché tutte le aree: gestione del testo, linguaggio di script, importazione ed esportazione dei dati ed altro ancora.
Questo l'indirizzo per scaricarli:
http://www2.filemaker.fr/italy/support/updaters.html.

INBox
L'esperto risponde...

File Htaccess in IIS

Buongiorno mitica redazione. Mi spiegate cosa accidenti sono questi file *.htaccess* che trovo dappertutto e che a prima vista sembrano non servire a un fico secco?

Risponde la Redazione

Grazie per il "mitici", birra e pizza come concordato? I file *.htaccess* sono di solito presenti in tutti i progetti nati per essere utilizzati con il Web Server Apache. *Htaccess* è infatti un file che consente di impostare tutte le impostazioni relative a un sito Web ospitato da un server Apache. Supponiamo per esempio di voler proteggere una cartella con una password. Il file *.htacess* debitamente configurato e per mezzo di appositi strumenti ve lo lascia fare. IIS non prevede l'uso del file *.htaccess*, la limitazione può essere tuttavia aggirata utilizzando alcuni tool commerciali come *IISPassword* disponibile all'indirizzo *www.troxo.com/products/iispassword/*

Debug di IIS

Gentile redazione, in determinate condizioni il mio IIS crasha. Non riesco a capire dove sia il problema. Le mie pagine sembrano funzionare perfettamente, probabilmente c'è qualcosa che non riesco a individuare in qualche pagina e non riesco a capire dove. C'è un modo per fare il debug del crash di IIS e capire dove sta l'inghippo?

Risponde la Redazione

Certamente. Esiste un tool: IISState disponibile all'indirizzo *www.iisfaq .com/Default.aspx?tabid=2513*. Il funzionamento non è complesso ma richiede un attimo di attenzione. L'installazione è semplicissima. Basta compattare il file in una directory e tutto funziona. Il secondo passo è rintracciare il PID del processo *inetinfo.exe* che è quello che corrisponde a IIS. Per farlo si può avviare il *Task Manager* con *CTRL+ALT +DEL* , entrare nella tabsheet dei processi, cercare *inetinfo.exe*. Il PID compare nella seconda colonna informativa della tabella visualizzata. Una volta ottenuto il numero di PID, è necessario avviare una console DOS, entrare nella cartella dove abbiamo scompattato *IISState* e digitare:

```
iisstate -p <PID> -d -hc –sc
```

In cui PID è il numero di PID che abbiamo ottenuto in precedenza. Con questo *IISState* si incollerà al processo in questione e intercetterà un eventuale crash dovuto a un crash hardware (Un crash tipico da Dr. Watson) o un crash Software (tipico ASP0115). Appena si verificherà una condizione di crash verrà generato un file di log e un file di dump. Il passo successivo è inviare il file di log sul newsgroup *microsoft.public.inetserver.iis*. Il newsgroup è supportato da Microsoft e un tecnico analizzerà il vostro log e vi dirà dov'è il problema. Naturalmente potreste leggere da soli il file di log, ma almeno le prime volte è consigliabile farsi aiutare. I newsgroup di supporto di Microsoft sono disponibili sul server *msnews.microsoft.com.*

Web Services

Cari amici di ioProgrammo. Ho un problema con i web services. In realtà ho capito tutto, cosa sono, come si usano e perchè sono cosi' potenti. Ho capito anche quela roba tipo UDDI. Ora, se volessi usarli in pratica, mi fornireste qualche link di registry UDDI?

Risponde la Redazione

Gentile amico, detto fatto.

http://www.xmethods.net
http://uddi.microsoft.com
http://www.salcentral.com
http://www.bindingpoint.com

Per gli altri che si affacciano solo adesso al mondo dei web services diciamo soltanto che in questi siti è possibile trovare una quantità sterminata di web services pronti per essere utilizzati nelle vostre applicazioni sia web sia standalone.

MySQL e le SubQuery

Gentili amici di ioProgrammo. Sto impazzendo per capire se e come sia possibile usare le subquery in MySQL. Il mio problema è il seguente, devo creare una query che mi restituisca l'insieme dei valori contenuti in una tabella e che ha almeno un record collegato in una tabella correlata. In pratica ho due tabelle. Nella prima ho uno UID che è un contatore unico che identifica ciascun record, nella seconda ho un campo UID ripetuto che è collegato alla prima tabella. Se nella seconda tabella non c'e' nessun record con quel particolare UID, la query non mi deve restituire record.

Risponde la Redazione

Caro lettore, per la salute dei suo neuroni: MySQL, dalla versione 4.1 supporta le SubQuery. In pratica è possibile, con una sola istruzione, eseguire una query sui valori restituiti da una sub query innestata. Ad esempio:

```
SELECT c.name,c.uid FROM USERS c where
      c.uid in (Select uid from NODE  where
                              type='image')
```

Questa query seleziona i valori *name* e *UID* dalla tabella *USERS* tale che i record della tabella *USERS* abbiano almeno un valore corrispondente nell'insieme dei record restituiti nella tabella *NODE.* In pratica viene prima effettuata la subquery *Select uid from NODE where type='image'* che restituisce tutti i record tali che *type* sia = *'image'.* Poi viene eseguita la prima query che preleva tutti i valori della tabella *USERS* che abbiano almeno uno UID corrispondente nell'insieme dei valori che abbiamo ottenuto in precedenza.
Sembrerebbe un po' complicato ma non lo è affatto. La tecnica è molto potente.

Provando lei stesso si renderà conto della sua praticità.

Inviare una form tramite un link

Non chiedetemi perché lo faccio. Da qualche tempo uso DreamWeaver per lo sviluppo delle mie pagine. Mi trovo benissimo fino a che non devo fare delle cose che sono al di fuori dall'automazione offerta dall'ambiente. Vengo al punto: ho una form, voglio ovviamente inviare il contenuto della form a uno script che lo gestisca. Il mio problema è che non voglio usare un bottone per azionare l'evento *submit*, ma un normalissimo link. Come posso fare?
Chiaramente non voglio inviare i dati nella linea di comando, perciò sono costretto a usare una form.

Risponde la Redazione

Una soluzione è quella di usare javascript nel seguente modo:

```
<form name="form2"  method="POST"
                    action="script..php">
```

il link sarà invece composto come segue:

```
<a href="#" onclick="eseguioperazione()">
                    Esegui Operazione</a>
```

e la funzione *eseguioperazione* sarà composta come segue:

```
function eseguioperazione()
{
  theform=document.getElementById('form2');
  theform.submit()
}
```

Questo ha chiaramente il vantaggio, tra le altre cose, che il link può essere posizionato ovunque nella pagina e non necessariamente all'interno della form.
Un piccolo svantaggio sta nell'utilizzo della *getElementByID* che, in particolari condizioni, potrebbe non referenziare l'oggetto corretto. Una soluzione è usare la funzione *MM_ findObj* così come viene creata in *DreamWeaver* e sostituire la seconda linea con:

```
{ var theform= eval(MM_findObj('form2')) }
```

Come trasferire dati da una pagina a un'altra in .NET

Utilizzo .NET da tre giorni e da tre giorni mi scontro con il più stupido dei problemi. Normalmente per passare i dati da una form su una pagina a uno script per una successiva elaborazione, avrei usato un normale POST o GET e un bellissimo button SUBMIT.
In .NET mi pare che non esista traccia di questa roba.
Come devo fare?

Risponde la Redazione

Per quanto possa sembrare strano, questo tipo di tecnica non è frequente ne consigliata in .NET. Tuttavia, ove proprio ce ne fosse bisogno, all'inizio della pagina è necessario inserire la direttiva:

```
<%@ Page Language="VB" ClassName=
                    "SendingPage" %>
```

Bisogna poi creare una proprietà per ciascun valore che si vuole passare a una seconda pagina:

```
Public ReadOnly Property GridData() As
                    System.Object
  Get
    Return DataGrid1.DataSource
  End Get
End Property
```

Infine per passare i dati:

```
Sub Button_Clicked(sender As Object, e
                    As EventArgs)
  Server.Transfer("ReceivingPage.aspx")
End Sub
```

Il numero di byte di una directory

Sono un giovane appassionato e programmo da pochi anni per diletto. Sto cercando di far funzionare una piccola utility per la gestione dei file, scritta in Visual Basic. Avrei bisogno di una funzione che mi restituisca la dimensione in byte di una directory: come posso fare? Vi ringrazio.

Giovanni Perfetti

Ciao Giovanni, di seguito trovi una funzione che realizza esattamente quanto indichi:

```
Function DirUsedBytes(ByVal dirName As _
                    String) As Long
Dim FileName As String
Dim FileSize As Currency
If Right$(dirName, 1) <> "\" Then
    dirName = dirName & "\"
Endif
FileSize = 0
FileName = Dir$(dirName & "*.*")
Do While FileName <> ""
    FileSize = FileSize + _
        FileLen(dirName & FileName)
    FileName = Dir$
Loop
DirUsedBytes = FileSize
End Function
```

Per interrogarla, è sufficiente passarle il nome della directory:

```
MsgBox DirUsedBytes("C:\Windows")
```

Validare un indirizzo mail

Gentile redazione vorrei conoscere un metodo, il più semplice possibile, per validare un indirizzo mail in una form HTML. Potreste darmi una mano?

Gianluca Pezzarossa

Gentile Gianluca, di seguito puoi trovare un codice in VBScripr che fa al caso tuo. Come noterai, utilizziamo le regular expression per stabilire la corretta forma di un indirizzo email:

```
<script language="vbscript">
function ValidateEmail(sEmail)
  set myExpression = new RegExp
  myExpression.pattern =
   "^(\w+\.)*(\w+)@(\w+\.)+([a-zA-Z]{2,4})$"
  If myExpression.test(sEmail.value) = True
Then
        msgbox "indirizzo e-mail valido"
  Else
        msgbox "indirizzo non valido"
  End If
End Function
</script>
```

PER CONTATTARCI

e-mail: ioprogrammo@edmaster.it
Posta: Edizioni Master,
Via Cesare Correnti, 1 - 20123 Milano

NEWSGROUP
Le informazioni nella Rete

ioProgrammo seleziona per voi le discussioni più interessanti dai newsgroup tecnici di programmazione. Se avete un problema molto probabilmente qualcuno l'ha già affrontato e risolto prima di voi

Java
it.comp.java

Installare nuove librerie di classi

Salve a tutto il newsgroup. Chi mi scioglie questa curiosità? Volevo sapere come scaricare nuovi package. Innanzitutto, mi sembra che si tratti di file con estensione .jar.Una volta scaricati, quale procedura si deve utilizzare per renderli disponibili al compilatore? Bisogna settare qualche impostazione? In quale cartella va messo questo file? Grazie mille. Saluti.

Postata da Agrick

Risponde Carlo Dainese

Ciao, l'uso di librerie di classi in formato *.jar* si effettua facendole vedere alla *jvm* via classpath: *java -cp miopath/mia-libreria.jar miaclasse.*
Con gli IDE, il principio ovviamente è lo stesso, ma agganci i jar via tool grafico sul singolo progetto java sul quale vuoi lavorare. Se invece metti i *jar* nelle *"/lib/ext"* del *jdk, jre* o di un *app.server* allora forzi l'utilizzo di quelle classi (le fai "vedere" alla *jvm*) per tutte le applicazioni che girano in quella *jvm.*

Dimensione dei caratteri

Ciao a tutti, la cosa sembra banale ma io non ho capito proprio come fare, qualcuno mi saprebbe spiegare come settare la dimensione dei caratteri? Magari utilizzando la classe Graphics2D. Grazie a chiunque mi risponda.

Postata da Andrea

La cosa è estremamente semplice. :-)) e non c'è bisogno di ricorrere a *Graphics2D*:

```
public void paint(Graphics g) {
...
   g.setFont( new Font("Serif",Font.PLAIN,12) );
   g.drawString("Stringa con carattere
                                        12",50,50);
...
}
```

Percorso di file in Java

**Ho un piccolo problema, devo inserire il percorso di un file in java, ho sempre usato la doppia barra, ma mi hanno consigliato di usare il percorso relativo, che non ho ben capito come funziona, qualcuno me lo può spiegare e farmi un esempio.
Vi ringrazio.**

Postata da Filippo

Rispone ReDiCuori

Un percorso assoluto è qualcosa come questo:

```
c:\programmi\mysql\bin\mysqld.exe
```

Un percorso relativo è qualcosa come questo:

```
mysql\bin\mysqld.exe
```

Nel primo caso indichi tutte le ramificazioni che devi percorre per arrivare al file *exe,* indicando la lista di tutte le directory che devi attraversare partendo da root (*c:*)
Nel secondo caso indichi solo una parte del percorso, in questo caso per default la directory di partenza è quella attualmente usata. Ad esempio, se ti trovi in *c:\documenti* il percorso relativo lo puoi vedere con il seguente percorso assoluto *c:\documenti\mysql\bin\mysqld.exe.*
Il percorso relativo è molto utile in casi come questo: hai creato un programma corredato del suo bel pacchetto di installazione. Il cliente installa il software nel HD *"D:"* e nella directory *programmi* creati da *filippo.* A questo punto se il tuo programma deve far riferimento ad un file *"sfondo.gif"* se utilizza un percorso assoluto *"c:\programmi\programmaFilippo\image\sfondo.gif",* non troverà mai il file mentre, se utilizza un percorso *"\image\sfondo.gif",* il file viene trovato senza alcun problema, in quanto considera come directory di partenza *"d:\programmi creati da filippo\.*

Parentesi graffa in netbeans

Ciao a tutti, ieri ho provato *NetBeans*, l'ho trovato molto buono ma non riesco a fare le parentesi graffe con *ALT+123* e *ALT+125*. Negli altri editor mi funziona tutto benissimo. C'è da dire che uso un portatile, quindi, negli altri editor, prima di fare quella combinazione devo premere il tasto *Fn*: purtroppo con *NetBeans* non funziona. Spero che mi sappiate aiutare perchè finalmente avevo trovato l'IDE gratuito adatto a me. Ciao e grazie per le eventuali risposte.

Postata da nav

Rispone Davide Consonni

Basta premere *shift + altgr + [* per la parentesi graffa aperta e *shift + altgr +]* per quella chiusa.

Visual Basic
it.comp.lang.visual-basic

Eseguire un exe esterno all'applicazione

Con il seguente codice eseguo un file exe esterno:

```
Dim r As Long
    r = Shell("C:\Programmi\Dir1\
            pippo.exe", vbNormalFocus)
```

Naturalmente se non viene più installato in C:\ ottengo un errore di file non trovato. Posso usare *$AppPath*? O risolvere in qualche altro modo il problema? Grazie.

Postato da Birra

Risponde DocDoc

Si, certo, che puoi utilizzare *AppPath*, ecco come:

```
r = Shell(App.Path & "\pippo.exe",
            vbNormalFocus)
```

Connessione a MySQL

Come posso connettermi a un db MySql che gira in remoto dalla mia applicazione sviluppata in vb (e la stessa in asp) utilizzando ADO? Grazie!

Postato da jak

Risponde Darkbyte

Installa i driver *MyODBC* e usa questa stringa di connessione:

```
Driver={MySQL ODBC 3.51 Driver};
Server=nome_o_ip_del_pc_server;
UID=tuo_userid;
PWD=tua_password;
Database=tuo_db;
OPTION=numero;
```

Per il parametro *OPTION* cerca sul sito di mysql (*www.mysql.com*), io uso 3. Ma ci sono più combinazioni in base alle tue necessità particolari...

NotifyIcon in VB.NET

Ho bisogno di più icone all'interno di un file eseguibile. Le icone devono essere utilizzare con notifyicon e il form principale è invisibile. Grazie.

Postato da Marco

Risponde Zanna

Puoi usare una imagelist oppure aggiungere le immagini al progetto e notificarle come risorse incorporate.
Ciao.

Web Services e Visual Basic

Salve a tutti, dovendo realizzare un WebService, ho fatto una ricerca sul Web, e sembra che non sia possibile crearlo in Vb, mentre su .NET ci sono un sacco di esempi sull'argomento. Prima di imbattermi in .NET, quindi, volevo sapere se qualche eletto è riuscito a realizzarlo in Vb 6. Grazie a chiunque vorrà rispondermi.

Postatto da Filippo

Risponde Andrea Saltarello

Puoi realizzare WS anche con VB6, utilizzando il *SOAP Toolkit* (che puoi scaricare dal sito MS). Fondamentalmente, ti permette di creare un wrapper che espone un componente COM come servizio web. Ciò premesso, realizzarli con .NET è decisamente preferibile per una (lunga) serie di ragioni.
Ciao.

Due eseguibili in uno

Salve, ho due file eseguibili e vorrei che il primo file potesse eseguire il secondo, il problema è banale se i due file sono separati (ad esempio nella stessa cartella) basta che il primo exe esegua una shell exe2. Esiste, però, un modo per creare un unico file e che quando si esegue esso lanci il secondo eseguibile inglobato in se stesso? Grazie a tutti.

Postato da Marco

Risponde Rafunk

Fai così: aggiungi al progetto del primo eseguibile un file di risorse che contiene il secondo EXE.
Con il *VB Resource Editor* aggiungi una risorsa custom e le dai un ID appropriato (il nome del secondo exe ad esempio).
In tal modo una volta che hai il secondo exe, puoi ricompilare il primo tutte le volte che vuoi e questo incorporerà sempre il secondo. Da codice estrai e lanci il secondo exe così:

```
Sub ExtractAndRun()
Dim bData() As Byte
    bData = LoadResData("prog2.exe",
                        "Custom")
    Open "prog2.exe" For Binary As 1
    Put #1, , bData
    Close 1

    Shell "prog2.exe"
End Sub
```

Formattare campi valuta

Ho diversi campi testo (su db access e via odbc). Nel database ho dei campi numerici in doppia precisione ma vorrei che venissero visualizzati in dei textbox come campi valuta. Cosa devo impostare per farlo?

Postato da Danilo

Risponde Albe V°

Ecco ciò che ti serve:

```
Textbox1.Text=Format$(Rs.Fields.Item(
            "Campo").Value, "0.00")
```

Connessioni lunghe e brevi

Secondo voi è meglio aprire una connessione a inizio programma e fare riferimento a essa, oppure aprire e chiuderla quando serve? Parlo di applicazioni tipo gestionale e simili... Voi nella maggior parte dei casi che fate?

Postato da Darkbyte

Risponde Andrea Raimondi

Dipende molto dall'architettura del database e dal tipo di connessione in oggetto. Supponendo che ci siano delle licenze per singolo client, trovo più logico che tu abbia una connessione solo quando serve. Qualora invece non ci fosse quel genere di restrizione, puoi provare a mantenere viva la connessione.
Ancora, però, ci sono diverse altre cose che vanno tenute in debito conto: il traffico di rete, la connessione ad internet(se l'applicazione gira sfruttando la Rete) ed altri fattori. In definitiva, direi che è sempre meglio connettersi quando c'è bisogno, a meno di esigenze specifiche.

PER CONTATTARCI

e-mail: ioprogrammo@edmaster.it
Posta: ioProgrammo - Edizioni Master, Via Ariberto, 24 - 20123 Milano

SOFTWARE SUL CD

Zend Studio 3.5

Un ambiente di sviluppo professionale, completo e multipiattaforma per PHP

Raramente ioProgrammo si dilunga nella presentazione di un tool commerciale. Nel caso di Zend siamo costretti a fare un'eccezione. Lo facciamo volentieri, perché al di la del valore commerciale del software, c'è un valore tecnico di eccezionale spessore.
Prima di ogni cosa c'e' da dire che Zend è la stessa azienda creatrice di Zend Engine I e II, ovvero il cuore pulsante di PHP 4.x prima e 5.x dopo. Già questa è una garanzia. Il nome da solo non basterebbe se poi l'applicazione non fosse dotata di funzionalità tali da rendere merito ai creatori di PHP. In particolare Zend Studio include ogni cosa per assistere un programmatore nel ciclo di sviluppo del proprio software. In particolare sono almeno tre gli aspetti da mettere in evidenza

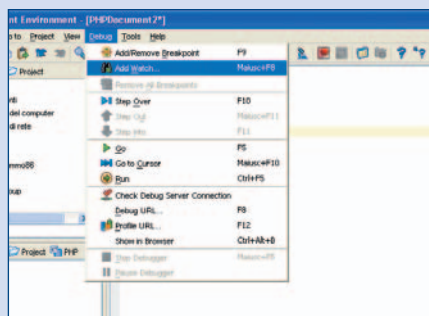
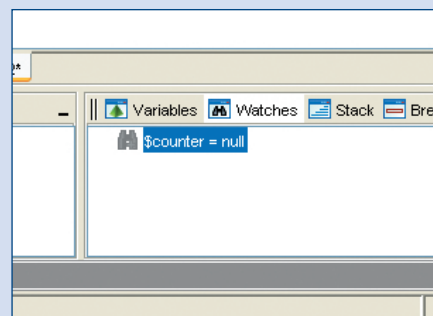
- **Zend IDE: dotato di code completion, syntax highlighting**, utilissima la funzione che evidenzia due paia di parentesi, quadre, graffe, tonde, corrispondenti e l'altra che mostra eventuali errori di digitazione a runtime. Inoltre interessante la funzione che mostra i tooltip per la referenzazione dei prototipi delle funzioni, anche quelle non insite nel linguaggio ma create dal programmatore. Si tratta, insomma, di un IDE potentissimo, che aumenterà a dismisura la velocità con cui svilupperete i vostri progetti PHP.

- **Zend Debugger**: All'interno del pacchetto è previsto un incredibile debugger, dotato di tutte le funzionalità normalmente tipiche di debugger per applicazioni standalone. Consente di inserire dei BreakPoint, delle Watch per i valori delle variabili e consente l'esecuzione step by step. Tutto ciò permette di individuare molto più facilmente errori in fase di sviluppo. Normalmente si otterrebbero gli stessi risultati solo inserendo

> ☑ **Zend Studio 3.5**
> **Produttore: Zend Corporation**
> **Sul web:** www.zend.com
> **Prezzo: $ 350**
> **Nel CD: /zend**
> ***Collegandosi a* www.zend.com/ioprogrammo *potrete acquistare il prodotto con uno sconto del 20% sul valore commerciale***
>
> **-20% solo per i lettori di ioProgrammo**

AVVIARE UNA SESSIONE DI DEBUGGING

```
<?
$counter=0;
while ($counter < 10000) {
        $counter++;
}
?>
```

1 Un piccolissimo esempio che utilizzeremo per mostrarvi come si esegue il debug di un'applicazione con Zend. Banalmente un ciclo che incrementa un contatore.

2 Dal menu *Debug* Selezioniamo "*Add Watch*". Con questa operazione diremo al debugger che vogliamo osservare come varia la variabile *counter* ciclo per ciclo.

3 La variabile *counter* appare adesso nella "*Debug Window*". E qui che dovremo guardare per capire come varia il suo contenuto durante il ciclo di vita dell'applicazione.

dei codici di controllo nel progetto PHP che andrebbero rimossi in fase di deployment.

- **Zend Server Center**: Installa tutto il necessario (eccetto *MySQL*) per provare le proprie applicazioni, in locale.
  In ambiente Windows utilizza la *Zend WinEnabler Technolog* per ottimizzare le prestazioni.

Di contorno a questi fondamentali, una serie sterminata di opzioni piu o meno interessanti che rendono molto rapida la programmazione in PHP, si va dal file inspector al CVS client integrato per lo sviluppo di progetti collaborativi.
Ultimo punto ma non meno importante:

Zend è scritto completamente in Java e gira praticamente su tutte le piattaforme più comuni:

Windows, Linux, Mac.

1. *File Manager* Consente di navigare nel proprio Hard Disk o in un eventuale FTP Server

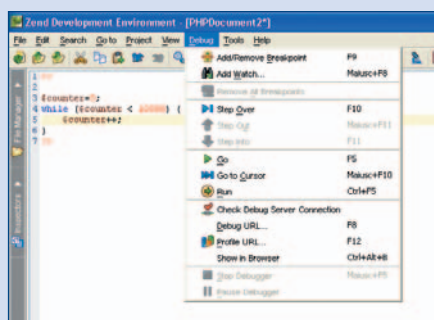
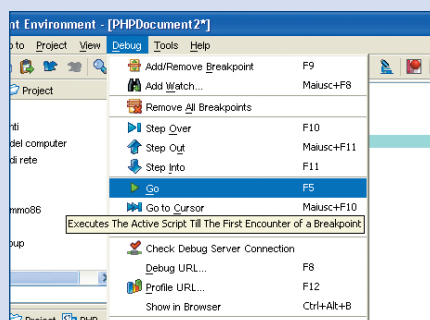
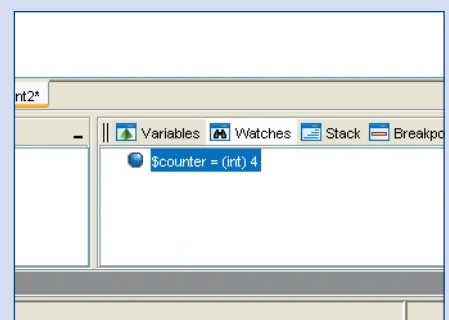
2. *Finestra del codice* Il vero e proprio editor, con tanto di code completion e sintax hilighting

3. *Debug Output* Mostra in formato HTML quello che dovrebbe essere l'output della nostra applicazione

4. *Debug Window* Qui dentro si può leggere il contenuto delle variabili "osservate" nella fase di debugging

5. *Tabbed Mode* È possibile spostarsi nei vari file aperti tramite delle comode tabsheet

6. *Debug Messages* Informazioni sullo stato d'esecuzione del debugger

7. *File Inspector* Per avere sotto controllo tutti i file coinvolti nel progetto

4 Posizioniamoci con il cursore sulla linea in cui vogliamo aggiungere un breakpoint, poi premiamo il tasto *F9*, oppure inseriamolo dal menu *debug*. In fase di debug l'applicazione si interromperà quando incontrerà un breakpoint

5 Attiviamo il debugger, tramite la pressione del tasto *F5* o dall'apposito menu. L'applicazione andrà in esecuzione e si stopperà non appena incontrerà un breakpoint.

6 Nella finestra di *Watch*, vedremo come varia il contenuto della variabile *counter* al susseguirsi dei cicli. Al primo breakpoint l'applicazione si stopperà e proseguirà alla successiva pressione del tasto F5.

Installer2Go 4.1

Un sistema facile e gratuito per realizzare pacchetti d'installazione

Un tool per la creazione di file autoinstallanti che non impegna lo sviluppatore nel dover imparare l'ennesimo linguaggio di scripting: con una interfaccia che punta tutto sul drag&drop, realizzare pacchetti di installazione diventa un vero gioco da ragazzi.

Fig. 1: *I tab consentono di passare rapidamente da una scheda all'altra*

Installer2Go va incontro a tutte linee guida per la certificazione Windows XP: è possibile utilizzare gratuitamente l'applicazione, a patto di accettare un piccolo messaggio pubblicitario che riporta alla home page del produttore di installer2go nei pacchetti di installazione che distribuiamo. L'interfaccia dell'applicazione si presenta come una maschera a tab che, al prezzo di un impatto visivo abbastanza complesso, consente di tenere sotto controllo l'intero progetto.

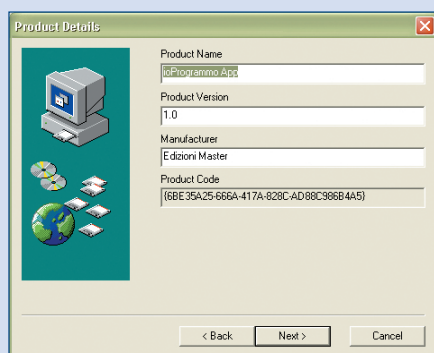
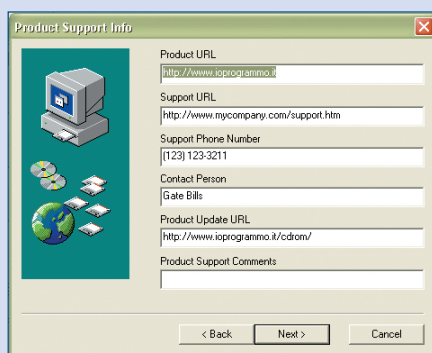
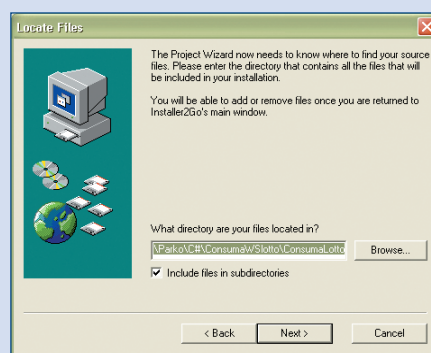
LA COSTRUZIONE DEL PROGETTO

La creazione di un nuovo progetto è assistita da un wizard che si attiva richiamando la voce *New Project* dal menu *File*. Definite tutte le informazioni relative ai dati "anagrafici" dell'applicazione si passa alla directory di destinazione. È quindi possibile, nella scheda *Requirements*, definire i requisiti minimi per l'installazione (come ad esempio la versione di Windows). Nella scheda *Files*, si indicano i file che appartengono al progetto, mentre in *Shortcuts* si definiscono icone e descrizioni.
Nella scheda *Registry*, si vanno a indicare i valori di registro e così via, fino al completamento delle varie schede. L'ultima, *Create Setup*, consente di creare il file di installazione vero e proprio.

Fig. 2: *Le impostazioni per le finestre di dialogo del setup*

Purtroppo, fra le tante lingue supportate manca proprio l'italiano: si spera che la lacuna sia colmata a breve.

☑ Installer2Go 4.1
Produttore: SDS Software
Sul web: www.dev4pc.com
Prezzo: $ 49,00
Nel CD: *i2g.exe*

IMPOSTARE IL PROGETTO CON IL WIZARD

1 **Il primo passo per un nuovo progetto è lanciare il wizard. Specifichiamo nome dell'applicazione e tutti i dettagli relativi all'autore e alla casa produttrice.**

2 **Si possono definire i recapiti Web dello sviluppatore o dell'azienda produttrice e tutte le informazioni relative al supporto post-vendita dell'applicazione.**

3 **L'ultima schermata del wizard ci invita a indicare la directory base del progetto: tutte le sottodirectory entreranno in quello che sarà il pacchetto di installazione.**

Apache

Le versioni aggiornate di uno dei Web Server più famosi al mondo

Con la versione 2.0.52 viene in particolare fissato un bug di sicurezza legato alla direttiva "Satisfy". Nelle versioni precedenti la 2.0.52, un hacker sfruttando questo tipo di falla avrebbe potuto bypassare la procedura di autenticazione ed entrare in zone riservate del sistema. Il bug era stato riscontrato nella versione 2.0.51 su tutte le piattaforme. Ref: http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0811. Anche la versione 1.3.31 viene rilasciata sulla base di bug fix legati alla sicurezza. In particolare:

- **È stato fissato un bug relativo alla scrittura di dati sul file di log**
- **È stato fissato un bug relativo alla gestione dei mutex che poteva provocare un blocco del server**
- **È stato fissato un bug relativo al parsing delle regole relative ai permessi d'accesso su indirizzi IP privi di una netmask.**
- **È stato fissato un bug relativo alla gestione del modulo digest per l'autenticazione criptata MD5.**

I riferimenti sono:

http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0987
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0020
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2004-0174
http://cve.mitre.org/cgi-bin/cvename.cgi?name=CAN-2003-0993

IN BREVE

JAVA

Text2gui Library 1.0

Tex2Gui rende semplice lo sviluppo di GUI basate su Java Swing. Si può sviluppare un'intera GUI descrivendola con una sintassi semplice, molto più semplice di come sarebbe se dovessimo utilizzare direttamente Java.

text2gui-1.0-eval.zip

Java Service Wrapper

Java Service Wrapper consente di creare le vostre applicazioni usando il linguaggio Java, e poi farle girare come servizio in ambiente Windows.

Javaservicewrapper\wrapper_win32_3.1.2.zip

JFrameBuilder 2.8

Un tool che semplifica la realizzazione di interfacce grafiche per applicazioni Java Swing. Attraverso un approccio drag&drop.

JFB_280.zip

Jedit 4.2

Se avete bisogno di un editor leggero per sviluppare le vostre applicazioni Java, eccolo qua!

\jedit\jedit42install.exe

Comersus Shopping Cart

Siete pronti per il commercio elettronico? Comersus è un'applicazione scritta in ASP che vi da una mano

Se avete necessità di sviluppare un sito di commercio elettronico senza dovere riscrivere da zero tutto il codice, Comersus è quello che fa per voi. Si tratta di un applicativo scritto in ASP che vi mette a disposizione un sistema completo di eCommerce pronto per essere personalizzato per voi e per i vostri clienti. E udite udite, quelli di Comersus dichiarano che il prodotto pur essendo scritto in ASP è compatibile con sistemi Linux.

\comersus

Schemester 1.1

Crea diagrammi Entità-Relazione e genera script

Una piccola e utilissima applicazione gratuita per la progettazione di diagrammi Entità-Relazione. È possibile esportare il progetto in DDL *(Data Description Language)*, consentendo così il dialogo con tutti i maggiori DBMS disponibili.

schemester.zip

TextPipe Pro 7.1

Una sofisticata applicazione per la manipolazione di testi

Una eccellente soluzione per quella vasta schiera di sviluppatori che vanno dai programmatori ai web-designer e che ha

Grafica 3D senza confini

Nel CD SPECIALE motori 3D

Accendi il miglior motore 3D open source!

IrrLicht è un motore per la grafica tridimensionale, scritto in C++ e utilizzabile sia con questo linguaggio, sia con i linguaggi di .NET. Presenta le principali caratteristiche che si trovano anche nei motori professionali e vanta una notevole comunità di sviluppatori, con diversi progetti in attivo.
IrrLicht ha tra i suoi pregi anche quello di poter utilizzare come librerie grafiche sia le Direct3D che le OpenGL, in maniera del tutto trasparente. Il funzionamento è inoltre garantito su diverse piattaforme operative tra le quali figurano tutti i Windows dal 98 a XP e Linux. Il supporto per MacOS è pianificato ma tuttora ancora non implementato.
Ad irlicht abbiamo dedicato un'intero articolo in questo stesso numero di ioProgrammo, lo trovate a pagina 32 della rivista.

irrlicht-0.7.zip

IN BREVE

FLASH

Gallery Component 1.3

Un componente che automatizza la costruzione di applicazioni per la realizzazione di gallerie di immagini.

\thumbnail gallery\GalleryComponent1_4a1.zip

FS Toolbar

Aggiunge una funzionalità per la gestione rapida della toolbar delle vostre applicazioni.

\toolbar component\fsToolbarFreeDistr.zip

MovieClip Tweenings

Aggiunge un metodo tween alle proprietà del filmato. Il metodo tween può essere usato per generare un animazione in modo programmatico.

MovieClip Tweenings\source.zip

Flash, cookie e ASP

Con Flash non è possibile settare direttamente e leggere i cookie. Con il file qui presentato questo limite viene superato.

FlashCookies.zip

PHP

PHP 5.0

La nuova versione con la sua rinnovata logica ad oggetti rappresenta un'opportunità da non perdere per chi sviluppa progetti di dimensioni notevoli.

\php50

Prado 1.5

È un framework che vi mette a disposizione una serie di funzioni che rendono il vostro linguaggio preferito Event Driven.

\prado\prado-1.5.zip

necessità di portare a termine complesse operazioni di manipolazione su file testuali. TextPipe include moltissimi filtri utili a ottimizzare l'aspetto del codice sorgente scritto in qualsiasi linguaggio. Sofisticato nelle funzionalità ma semplice da utilizzare. In questa nuova versione è stato rinnovato il Wizard che guida alla ricerca testuale all'interno di tag HTML: una volta trovato, il testo può essere cancellato o editato. Al primo avvio, è richiesto l'inserimento dell'indirizzo e-mail presso cui sarà inviata la chiave di attivazione per valutare il prodotto. Trenta giorni di prova.

textpipepro2.exe

SuperEdi 3.4.1

Un editor piccolo e funzionale per gli sviluppatori

Ideato appositamente per gli sviluppatori, SuperEdi può essere utilizzato sia per lo sviluppo in locale che per modificare file in remoto. Completamente gratuito, presenta tutte le principali funzionalità degli editor più blasonati: evidenziazione sintattica per i maggiori linguaggi, filtri per la manipolazione automatica del testo e supporto multilingua.

SuperEdi-3.4.1.U.exe

Crystal Flow for C 1.15

Per "familiarizzare" con il codice C

Spesso, anche se il codice l'abbiamo scritto noi, capire il funzionamento di un'applicazione può essere veramente arduo. Crystal Flow si occupa di generare il flow chart relativo alla porzione di codice che indichiamo, velocizzando notevolmente il tempo necessario a comprenderne la struttura.

Versione dimostrativa.

CrystalCFLOW_EvalADv115.zip

Database Tour 4.9.7

Gestisci i tuoi database con mille funzioni

Un tool per l'amministrazione dei database che si interfaccia a tutti i più diffusi DBMS presenti sul mercato. Consente di visualizzare e modificare i dati immagazzinati, oltre a fornire un avanzato sistema per la progettazione dei report. Testato su Paradox, dBase, FoxPro, ASCII, InterBase, Oracle, SQL Server, MS Access. Versione di valutazione valida 24 giorni.

dbtour.zip

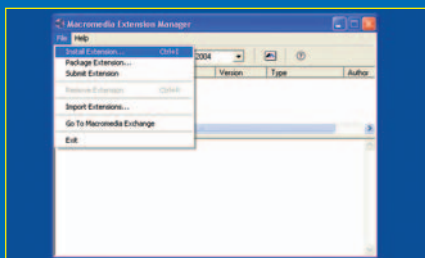
Stylus Studio XML Professional Edition 6.0 build 212

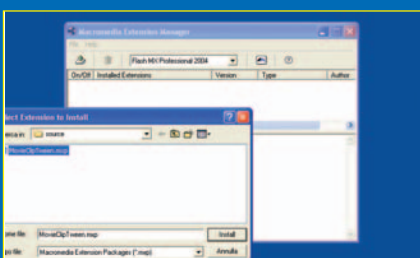
Gestisci XSL e XML con stile

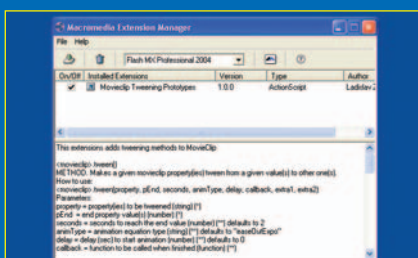
Forte dei molti premi vinti e di una base di installato che conta oltre 30.000 sviluppatori sparsi in tutto il mondo, Stylus Studio si presenta come un potente tool visuale per lo sviluppo e la gestione di documenti XML e XSL. Rimarchevole la qualità della mappatura dei documenti che Stylus Studio offre per via grafica, consentendo una più facile analisi dei documenti. Permette di gestire e modificare tutti i tipi di file che appartengono ad un'applicazione XML: oltre a documenti XML, abbiamo il pieno supporto per gli XSLT stylesheets, DTD e gli XML schema, oltre ai file Java. Gira su Windows NT – 2000 -XP.

Tra le funzioni più interessanti, segnaliamo la possibilità di effettuare il debug e la presenza di una potente funzione di pre-

COME INSTALLARE UN COMPONENTE FLASH

Apriamo l'extension Manager

1 Dal menù start di Windows/Programmi/Macromedia lanciare il macromedia extension manager. Dal menù file selezionare "Install Extension"

Selezioniamo il componente

2 Dalla finestra di dialogo di selezione dei file portarsi nella cartella che contiene il componente da installare e selezionarlo.

Verifichiamo il tutto

3 Se l'installazione è andata a buon fine, vedremo il componente nell'Extension Manager. Al primo riavvio di flash, sarà disponibile.

view sulle trasformazioni realizzate. Un prodotto che può risultare valido sia ai principianti che agli esperti e che si presta a fare da guida in un percorso di apprendimento delle tecnologie legate a XML. All'atto dell'installazione è richiesto un collegamento Internet al fine di riempire una breve form. Una chiave di attivazione verrà inviata alla nostra casella di posta elettronica.

studio.exe

Absolute Log Analyzer Standard 2.38

Recupera e analizza le informazioni sugli utenti del tuo sito

Un tool che consente l'analisi del traffico generato da siti anche di grandi dimensioni. Sono disponibili oltre 160 report diversi ed è possibile una fine regolazione dei filtri per il parsing dei dati. Le informazioni sono raccolte in un database che permette una agevole consultazione ex post. La nuova versione include la rilevazione dei googlebot (gli agenti di google) e dei più diffusi spider. Versione di prova valida ventuno giorni.

abslogdemo_setup.exe

PowerHEX 1.0

Advanced o Lite: scegli il tuo editor

Un tool che permette l'editing esadecimale sia del contenuto dell'hard disk sia della Ram. Dotato di un completo convertitore esadecimale, PowerHEX offre anche la possibilità di analizzare i pacchetti in transito sulla rete. Al momento dell'installazione è possibile scegliere fra due versioni: Lite per chi si avvicina al disassemblaggio, Pro per chi vuole sfruttare le funzioni più avanzate. Versione dimostativa valida ventuno giorni.

PowerHEXSetup.exe

RS232 Hex Com Tool 5.0

Per comunicare con qualsiasi periferica RS 232

Un terminale per comunicare, via RS 232,

IN BREVE

PHP Eclipse

Se PHP è uno dei linguaggi più diffusi del Web, Eclipse sta diventando uno degli editor preferiti dai programmatori. PHP Eclipse consente di personalizzare Eclipse come editor per PHP.

Phpeclipse\PHPEclipse1.1.0-2004-09-25.zip

Top PHP Studio v1.22

Un editor per PHP. Molto leggero, dotato di syntax highlighting. Installa un web server sulla vostra macchina, tale da potere provare i vostri script.

\tphptopstudio\tphpstudio-trial-en.zip

PHTML Encoder 3.8

Ecco a voi un enconder che vi consente di codificare i vostri script prima di distribuirli.

\phtmlencoder\phtmlenc38.zip

DBQWIKSITE 3.0

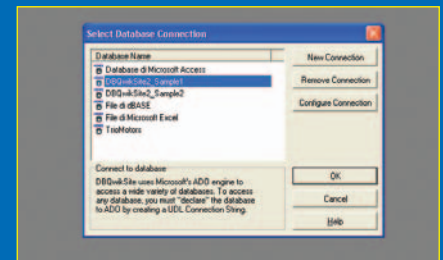
Volete creare un'applicazione collegata a un database in 5 minuti? dbQwikSite vi da una mano

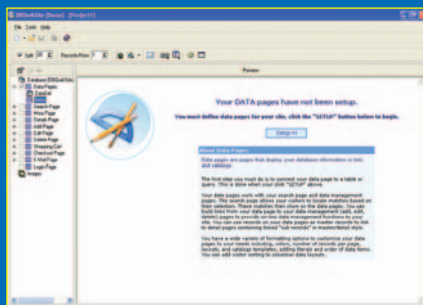
Si tratta di un generatore rapido di applicazioni per la gestione dei database. In sostanza è sufficiente "dargli in pasto" un database, impostare poche opzioni, premere un bottoncino, e la nostra interfaccia PHP o ASP verso una base di dati sarà pronta in un battibaleno.

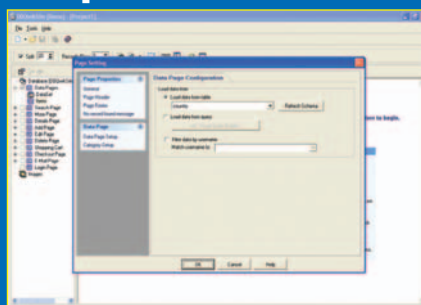
\dbquicksite\dbqwiksitepro.zip

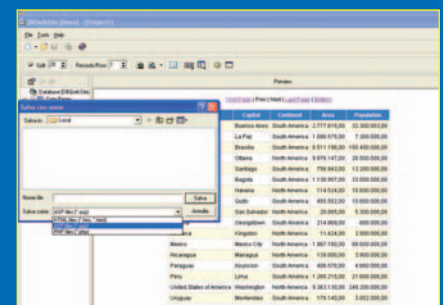
Creiamo un nuovo progetto

1 Dopo avere lanciato il programma, la prima cosa da fare è selezionare "*Blank Project*" dalla maschera che compare e poi cliccare su "*Ok*".

Selezioniamo il database

2 La fonte di dati può essere un database Access, ODBC, o un database MySQL. In questo caso stiamo utilizzando uno dei database di esempio.

Selezioniamo la tabella

3 A sinistra nel menù ad albero selezioniamo "*Items*" dalla collezione "*DataPages*". E nella pagina centrale clicchiamo su setup.

Recuperiamo i dati

4 Selezionando "*Country*" in questa maschera, poi cliccando su "*Ok*". Vogliamo che i nostri dati siano prelevati dalla tabella "*Country*" del database.

Pubblichiamo

5 La tabella può essere ancora molto personalizzata. Quando avrete finito è possibile salvare il codice direttamente in formato ASP, o PHP.

IN BREVE

.NET

NCTDiscWriter ActiveX DLL 2.5

NCTDisc Writer è pienamente compatibile con ISO9660, consente di scrivere in modalità Joliet o multi-sessione sia su DVD che su CD. Consente di creare AudioCD partendo da MP3, MP2, OGG, WMA.

NCTDiscWriter2.exe

BarCodeWiz Barcode ActiveX Control 1.67

Questo ActiveX vi consente di utilizzare i codici a barre senza troppi problemi all'interno delle applicazioni che state sviluppando.

BarCodeWiz_BC_Activex_Demo.exe

Active Sound Studio 2.0

Consente di aggiungere effetti multimediali. Si possono applicare effetti speciali, gestire le playlist, ricercare le informazione nei tag associati all'audio.

amewp_t.exe

PureASP file upload v1.3

PureASP Upload è un componente per ASP che resolve il problema dell'upload di un file da remoto.

\pureAspUploader\pASPUpl2.zip

Regex Designer

RG garantisce un uso semplificato delle regular expression, usate per effettuare delle ricerche estremamente complesse su file di testo in modo del tutto programmatico.

\regexdesigner\RegexDesigner.zip

RssFeed

RssFeed è un componente .NET disegnato per importare gli ormai famosissimi file RSS nelle vostre pagine .NET

\rssFeeds\skmRss[1].1.5.zip

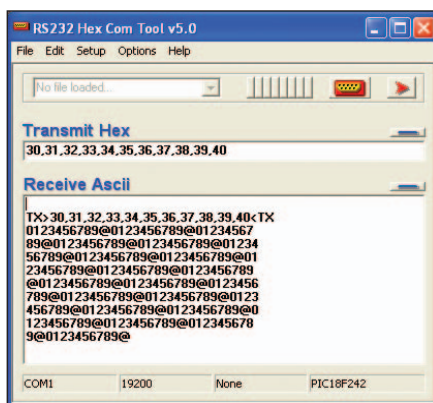
Sharp ZipLib

La libreria più famosa in ambiente .NET per la gestione dei file compressi. Si tratta di un must da mettere nelle proprie risorse.

\sharpziplib

Npersist

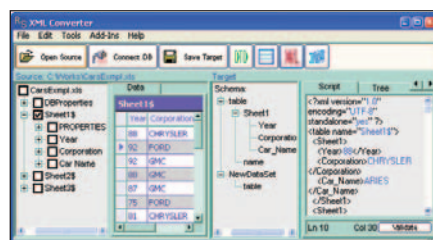
È un framework che vi mette a disposizioni una serie di strumenti per rendere meno problematica la gestione della persistenza dei dati.

\npersist\ NPersist-1[1].0.3.zip

con qualsiasi periferica. Interessante la possibilità di salvare i dati di setup per ogni periferica, potendoli richiamare al momento opportuno. I dati possono essere trasmessi e ricevuti sia in esadecimale che in ASCII. La velocità cui è possibile settare la trasmissione è compresa fra 110 e 115200 byte. Versione di prova valida trenta giorni.

RS232HD50.exe

XML Converter Professional Edition 4.28

Convertire i database in XML

Utile ed efficace questo tool che, collegandosi a tutti i più diffusi database, consente di convertire i dati in formato XML. È presente il supporto per ODBC, SQL Server, Oracle, MySQL e per i documenti Office. Una volta definito un template di riferimento, i dati possono essere riversati in

documenti XML anche in modalità batch: opzione fondamentale con basi di dati di grandi dimensioni. Versione dimostrativa.

XMLConverterPro.zip

XmlPad 1.0.0.1

Editare e formattare documenti XML

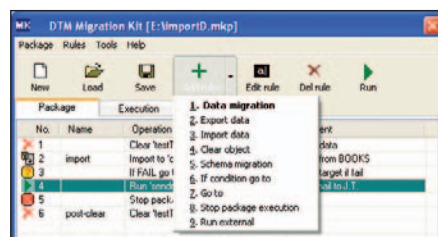
Piccolo e potente, questo editor è stato pensato per gli sviluppatori che vogliano gestire documenti XML senza troppi "fronzoli" e senza dover rinunciare alle funzioni più utili come la colorazione sintattica ed un efficace Wizard per l'autocompletamento del codice. Gratuito.

XmlPad.zip

DTM Migration Kit 1.02.06

Migrare fra differenti database

Per chi si trova nelle condizioni di dover spostare grosse quantità di dati da un data-

base all'altro, DTM Migration Kit fornisce tutte le più importanti funzioni per l'import e l'export verso più fonti di dati (ODBC, IDAPI or Oracle). Il processo di migrazione è completamente automatico e risultano supportati tutti i più diffusi DBMS. Versione dimostrativa, alcune stringhe sono sostituite dalla scritta "DEMO".

mk_demo.exe

HWDirect 1.05.00.70

Leggi e modifica i registri di sistema

Un avanzato sistema di interrogazione a

REGULAR EXPRESSION QUESTE SCONOSCIUTE

Sono croce e delizia dei programmatori. Croce perché la loro sintassi è veramente complicata, delizia perché sono estremamente potenti. La loro funzione è quella di ricercare una stringa all'interno di un testo che soddisfi un particolare criterio di ricerca (Pattern). Ad esempio:

```
Dim emailregex As Regex = New Regex("(?<user>[^@]+)@(?<host>.+)");
Dim ismatch As Boolean = emailregex.IsMatch("johndoe@tempuri.org");
```

Questa linea inizializza un oggetto di tipo espressione regolare. Viene poi richiamato il metodo IsMatch passandogli come parametro un indirizzo email. L'oggetto di classe Regex verifica che sia realmente un indirizzo email e per farlo usa l'espressione regolare dichiarata come argomento di Regex.

I contenuti del libro

RIVISTA + LIBRO + CD-ROM in edicola

AUDIO

Clip Sonori
- Le caratteristiche tecniche
- Concetti di base
- Elementi essenziali
- Riproduzione di un clip
- Riproduzione di un loop
- Elaborazione di un suono
- Acquisizione

Midi
- Introduzione a Midi
- Midi in JavaSound
- File MIDI
- Dispositivi MIDI
- Riproduzione di un file MIDI
- Suonare una nota
- All'interno del sintetizzatore

GRAFICA ED IMMAGINI

Grafica di base
- Cenni storici
- Arriva Java2D
- Sistemi di coordinate
- Processo di presentazione (rendering)
- Disegnare figure
- Gestione del tratto
- Riempire le figure
- Trasformazioni
- Composizione

Font e testo
- Un pieno di font
- Tutto sui font
- Font metriche
- Creazione di font

Immagini
- Un ventaglio di scelte
- Modello produttore/ consumatore
- Modello immediato
- Scrittura di immagini
- Formati supportati
- Creare immagini
- All'interno delle immagini
- Disegnare immagini
- Elaborazione di immagini
- Immagini e prestazioni
- Immagini volatili

Colori
- Colori in Java2D
- Spazi di colore
- Modelli di colore
- Colori ed immagini

VIDEO

Il supporto multimediale
- Ingresso, elaborazione e riproduzione
- Spazi di colore
- Streaming
- Presentazione
- Elaborazione
- Introduzione a JMF

Filmati
- Riproduzione
- Pannello di controllo
- Controllo del volume
- Visualizzazione del tempo trascorso
- Controllare la riproduzione
- Riorganizzazione e posizionamento
- Eventi

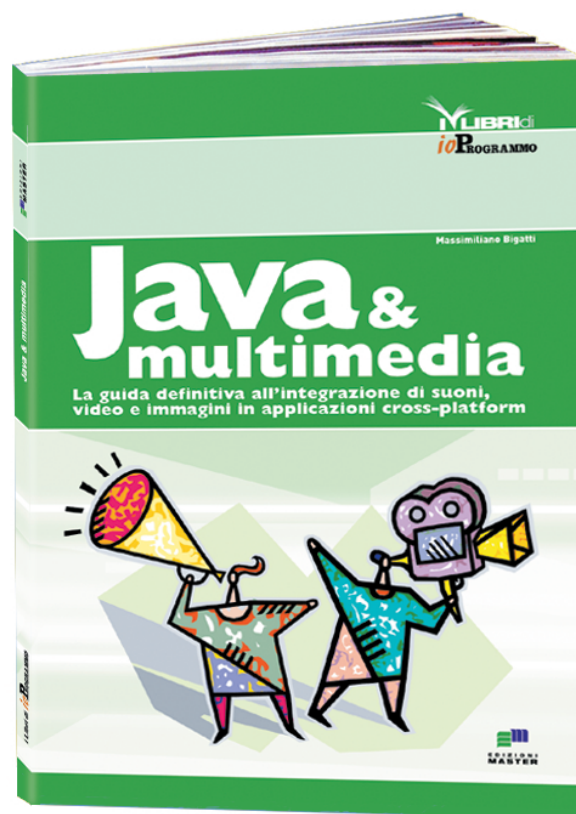
Effetti, memorizzazione ed acquisizione
- Effetto Letter Box
- Effetto bianco e nero
- Effetto Seppia
- Effetto controllo colore
- La base di tutti gli effetti
- Memorizzazione

Acquisizione
- Una rapida panoramica
- Dispositivi ed acquisizione

APPENDICI

Streaming
- RTP
- RTP e JMF
- Ricezione

Java Advanced Imaging
- Caratteristiche e funzionalità
- Operatori

3D con Java
- Obiettivi e caratteristiche
- Modello di programmazione

Java & multimedia

La guida definitiva all'integrazione di suoni, video e immagini in applicazioni cross-platform

Un libro che ci mostra una parte di java tanto potente quanto sottoutilizzata, ovvero la sua capacità di gestione degli oggetti multimediali. Nell'ottica di Sun, Java non è solo un potente linguaggio dedicato al mondo business, ma una piattaforma che abbraccia a 360° l'intero mondo dello sviluppo. Se da un lato siamo abituati a considerare Java per le sua API così complete e potenti, dall'altro lato non si tiene abbastanza in conto la sua caratteristica base e cioè il fatto di essere un linguaggio multipiattaforma. In questo senso Java2D, Java3D, Java Advanced Imaging e JMF rappresentano una straordinaria opportunità di creare software capace di conquistare ogni tipologia di mercato. Si va dal consumer dei player mp3 a potenti strumenti per l'elaborazione cinematografica, niente è precluso a Java. E' in questo spazio fatto di immagini, suoni e filmati che questo libro ci proietta. Il perfetto equilibrio fra la disquisizione teorica e la quantità di esempi messi a disposizione ne fa una guida utilissima che insegna come sviluppare in modo sapiente applicazioni multimediali e multipiattaforma con Java.

IN BREVE

Audio Capture ActiveX Control 1.0 pop
Consente di inserire nelle tue applicazioni un controllo per catturare l'audio da device di input differenti.

audiocapturedemosetup.exe

Asplib Component Library
Si tratta di una libreria di ben 16 componenti aggiuntivi per rendere ancora più rapido lo sviluppo di applicazioni in ambiente .NET.

Asplib\ AspLibSetup.msi

Aspose Component
Una serie di componenti che consentono ad applicazioni .NET di interagire facilmente con i programmi di Microsoft Office. Si va da Excel a Word a PowerPoint.

\Aspose

EaseSoft ASP.NET Barcode
Un componente in grado di facilitare la gestione di codici a barre all'interno di applicazioni .NET

\barcode control\easesoftlinear.zip

DotNetWebComponents
Un pacchetto di tre componenti per .NET. Un fantastico carrello per le vostre applicazioni di ecommerce. Una chat e un forum.

\dotnetwebcomponents\
DotNetWebComponents.exe

FCK Editor 2.0 Beta 2
Un editor visuale da inserire nelle vostre pagine web. Per semplificare l'immissione del testo in applicazioni di tipo CMS.

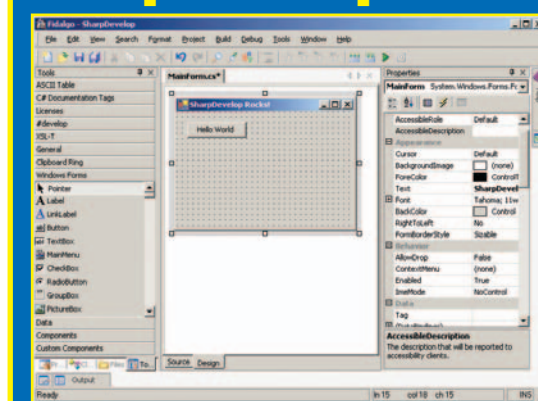
\fckeditor_20b2\FCKeditor_2.0b2.zip

Extended DataGrid
Un componente per .NET che aggiunge molte funzionalità al componenente di default *Windows.Forms, DataGrid.*

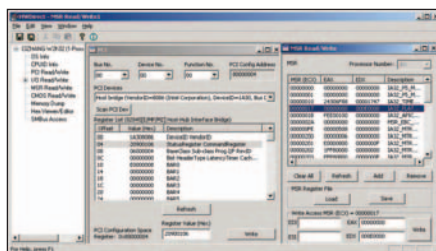
\extended datagrid\Leadit[1].
ExtendedDataGrid_0_1_2_2.zip

ASP
Barcode
Oltre venti tipologie differenti di codifica a barre e la gestione completa dei colori, dei font, di eventuali bordi e di tutto quanto riguarda l'aspetto grafico.

BarcodeASP.zip

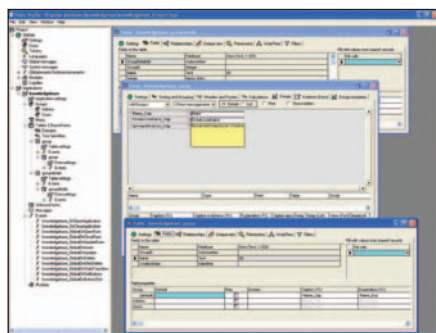
Sharp Develop 1.0.1

Chi pensa che Visual Studio o Web Matrix siano le uniche possibilità per programmare in .NET si sbaglia. Esiste anche questo interessantissimo Sharp Develop che oltre ad essere un IDE fatto straordinariamente bene, ha il grosso vantaggio di funzionare sia in ambiente Windows su Microsoft .NET che in ambiente Linux su Mono.

\sharpdevelop

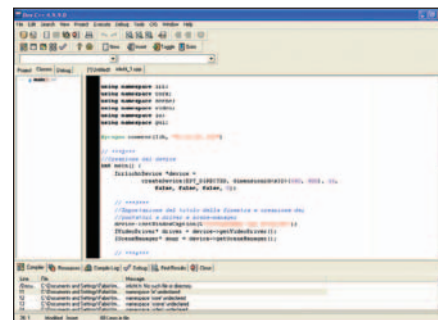
basso livello dei registri di sistema. Consente agli utenti di leggere e modificate tutti i registri hardware presenti nel sistema attraverso un'interfaccia che divide in otto gruppi le interrogazioni possibili: sistema operativo, CPU, PCI, I/O, CMOS sono i principali.

Versione dimostrativa, risulta disabilitata la funzione di scrittura.

hwd_inst.exe

Polar Studio 5.2
Costruisci applicazioni Web a partire da un database Access

Un interessante tool che consente di esporre come applicazione Web un qualsiasi database Access. I significativi esempi e l'efficace help online permettono di raggiungere buoni risultati in poco tempo. Il codice che fa da ponte fra il l'applicazione Web ed il database è ampiamente personalizzabile dall'utente. È possibile effettuare dei test in locale grazie alla presenza di un apposto Web server. Versione di valutazione, limitata nelle funzionalità. Al primo avvio è richiesto l'indirizzo mail dell'utente: in pochi istanti sarà inviata una chiave di attivazione

SetupPolarStudio.exe

Dev-C++ 5.0 beta 9
Chi lo dice che un IDE debba essere per forza molto pesante? Dev-C++ è leggero e molto potente

Bel IDE per lo sviluppo di applicazioni C++. Fra le caratteristiche più interessanti, debugger integrato. Class Browser, Code completion, ovviamente Syntax Highlighting. La versione che vi proponiamo è completa del compilatore *Mingw*, quindi pronta all'uso.

\devcpp\devcpp4990setup.exe

DBManager Professional 2.3.0 Freeware
Un software standalone per la gestione di database MySQL. Molto potente!

Uno dei grandi difetti di MySQL fino a qualche tempo fa era di non disporre di software di amministrazione remoti suffi-

cientemente flessibili. Recentemente la situazione è molto cambiata e anche MySQL dispone di una nutrita schiera di tools del genere. A dire il vero DBManager è uno degli strumenti storici per la gestione remota di database MySQL. Ha avuto perciò il tempo di diventare uno strumento maturo e flessibile. Al momento offre soluzioni piuttosto avanzate: dal query builder alla gestione dei metadati.

\dbtools\prosetup230EN.exe

Agast

Chi non ha mai provato a sviluppare un videogioco scagli la prima pietra. Con Agast questo non sarà più un problema

Non sempre rapido vuol dire anche potente e flessibile. In questo caso non è proprio così. Agast è un tool rapido per lo sviluppo di videogame in stile Monkey Island.
È semplicissimo da usare, ma anche molto potente. Assolutamente da provare.

Agast\agast.zip

Chalet 1.0

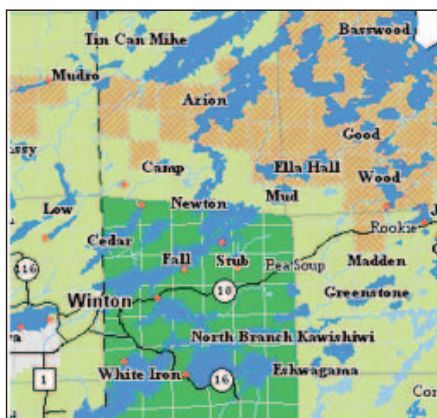
Vuoi inserire una chat IRC nelle tue pagine web? Chalet ti risolve il problema.

Chalet è un gateway scritto in java verso IRC. In pratica vi consente di sviluppare, oppure utilizzare direttamente una Java Chat con cui connettervi al meraviglioso mondo di IRC.

Chalet\chatlet-1.0.zip

MapServer 4.2.4

Pensavate che applicazioni per la gestione di mappe geografiche fossero possibili solo spendendo molti soldi? Ecco a voi MapServer, potente e gratuito

Allora, tenetevi forte, perché questo è un

tool assolutamente straordinario. Utilizzando questo fantastico MapServer sarete in grado di realizzare applicazioni per il disegno vettoriale di Mappe. Ok Ok, no avete ancora capito bene a cosa serve. Ve lo spiego un po' meglio. Prendete un'enorme mappa vettoriale della vostra città, datela in pasto a MapServer e sarete in grado di zoommare, rintracciare percorsi, calcolare distanze. Poi prendete una mappa degli alberghi della vostra città, sovrapponetela alla precedente, e sarete in grado di accendere o spengere il layer che più preferite, oltre tutte le cose che già potevate fare. E così via. Se avete pazienza con questo prodotto si possono sviluppare applicazioni che supportano mappe geografica altamente professionali.

\mapserver

Mono 1.0.2

Il porting del famoso .NET di Microsoft in ambiente Linux

Ok, che ci crediate o no .NET è uno standard. Perciò Microsoft potrebbe non essere l'unica a sviluppare un framework per .NET. Perciò ecco a voi Mono. Progetto ambizioso che si propone di ricalcare le gesta del più noto Microsoft .NET. Fun-

PERCHÉ USARE MONO?

Linux prima di diventare un sistema interessante dal punto di vista Desktop è stato interessante dal punto di vista Server. Tanto che una buona parte di quella che è l'Internet che conosciamo gira su sistemi Linux. E' anche vero che recentemente Microsoft sta rosicchiando posizioni nel mercato server ai sistemi Unix. L'introduzione di una tecnologia come .NET che rende semplificata la curva di apprendimento si è contrapposta a tecnologie preesistenti quali Java, Python, Perl, tipicamente appannaggio dei sistemi Unix ma che hanno una curva di apprendimento molto lenta. Cosi' Novell, che di recente ha acquistato Suse, ha deciso di investire sul progetto Mono che essendo multipiattaforma Linux/Windows ed essendo compatibile con gli standard .NET garantisce un erosione del mercato server più lenta. Mono funziona tranquillamente anche in ambiente Windows. Tanto che è utile per un programmatore .NET sviluppare in piena compatibilità, di modo che la propri applicazione possa girare su tutti i sistemi indipendentemente da quali siano le scelte di un eventuale sistemista o cliente.

IN BREVE

DataMatrix Barcode

Le immagini dei codici a barre prodotte sono orientabili lungo le quattro direzioni ed inoltre sono pienamente supportate la *Reed Solomon Error Correction.*

DataMatrixAsp.zip

File e directory

Della programmazione che fu, agli albori dell'era visuale, qualcuno ricorderà le difficoltà di accesso al file system. Una classe del .NET Framework oggi rende quelle stesse operazioni facili come non mai.

FileDir.zip

Immagini sconosciute

La manipolazione delle immagini via Web è da sempre in cima ai pensieri dei Webmaster. Il componente che presentiamo consente l'upload di immagini nelle vostre pagine Web con ampie possibilità di gestsione di tutte le proprietà dell'immagine stessa.

ImageCounter.zip

Upload più facile

Un controllo HTML che vi farà dimenticare delle difficoltà incontrate nel voler implementare una operazione così semplice con ASP senza l'utilizzo di componenti esterni.

Upload.zip

Bokai BarCode

Bokai barcode image generator .Net control è un componente .Net che ci darà una mano a generare codici a barre.

Bokaibarcode.zip

Strumenti di posta

Una suite per inviare e ricevere e-mail ed un mese di tempo (tanto dura il periodo di prova) per decidere se è questo il tool che vi serviva.

Toolsmail.exe

C#

Active Q

Se state pensando ad implentare una newsletter nelle vostre pagine web, allora dovete provare Active Q

Activeq.zip

Amministrazione Remota

Ecco un tool di amministrazione remota semplice ma completo, che risulta compatibile con qualsiasi browser.

DataAdmin.msi

IN BREVE

Equation Solver
Con questo componente avrete a disposizione le più raffinate procedure per la risoluzione di equazioni ad una variabile.

EquationSolver.msi

ESB Decimals
Tanti demo scritti in C#, VB.NET e Borland Delphi 8 vi saranno di grande aiuto per capire come utilizzare queste routine che per alcuni si riveleranno insostituibili.

EsbDecimals.zip

Interpolazione
Una raccolta di algoritmi per l'interpolazione lineare includenti il metodo polinomiale di Newton, la formula di Lagrange, l'algoritmo di Burlish-Stoer, le chiavi cubiche e l'interpolazione bicubica.

Interpolation.msi

Mail per .NET
La tecnologia bi-direzionale renderà facile creare, inviare e ricevere messaggi (secondo i protocolli POP, SMTP, IMAP).

Mail.net.msi

NewsLetter facile
Gestione della configurazione, dell'autenticazione dell'amministratore, la gestione completa degli iscritti.

Newsletter.zip

Creare screensaver
Un sorgente che implementa un semplice salvaschermo; non ha molto valore concreto, ma credo ne possa avere dal punto di vista formativo.

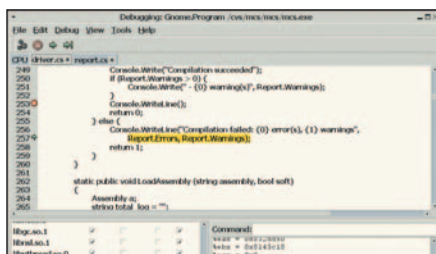
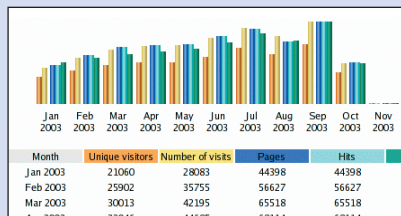
SSSource.zip

WebGrid
Un controllo per il web sviluppato in C#, con un layout completamente personalizzabile (specie per la visualizzazione dei record).

Webgrid.zip

WebStats
Registrare e consultare le statistiche relative ad un sito web pubblicato su un server che supporta il .NET Framework è ora a portata di tutti.

Webstats.zip

Month	Unique visitors	Number of visits	Pages	Hits
Jan 2003	21060	28083	44398	44398
Feb 2003	25902	35755	56627	56627
Mar 2003	30013	42195	65518	65518

ziona sia in ambiente Linux, che in ambiente Windows. Non è forse ancora maturo come il suo celebre gemello, però è sicuramente multipiattaforma, il che lo rende un prodotto da seguire con attenzione. La presenza di un tool in standard .NET sia per Windows che per Linux fa si che .NET diventi un acerrimo concorrente di Java che ai tempi della sua uscita si dichiarò rivoluzionario proprio perché Multipiattaforma.

Mono

OSCommerce 2.2

Uno dei migliori tool per la realizzazione rapida di siti dedicati al commercio elettronico

Nel tempo è diventata l'applicazione più usata per lo sviluppo di siti di commercio elettronico. È scritta in PHP, conta innumerevoli installazioni, esistono un'infinità di add-on per risolvere quasi ogni esigenza. Ha ancora il difetto di essere non facilmente personalizzabile. Tuttavia se volete mettere in piedi un negozio di commercio elettronico senza troppi problemi OSCommerce è quello che fa per voi.

\oscommerce\oscommerce-2.2ms2.zip

Tomcat 5.5.

Assolutamente indispensabile per chi vuole iniziare a programmare in JSP

E' l'application server essenziale per lo sviluppo di pagine JSP. JSP è una piattaforma decisamente interessante per chi ha già esperienza di programmazione java.

\tomcat5.5

IN BREVE

VB.NET

AChart
Componente dall'utilità indiscutibile, vi permetterà di generare fino a 21 diversi tipologie di grafici di base, creare effetti tridimensionali, gestire bordi, trasparenze, antialiasing, gradienti, etc.

Achart.msi

AExcel
Si possono creare fogli elettronici che supportano formattazione, funzioni, immagini, diagrammi; il tutto senza utilizzare Excel.

Aexcel.msi

ARecurrence
Per non dimenticare una ricorrenza o una data particolare, avete bisogno del componente .NET qui presentato.

Arecurrence.msi

ASpell
Ecco il componente multilingua che vi serviva: potrete controllare fino a venti linguaggi differenti.

Aspell.msi

ASPFTP
Potrete effettuare l'ftp direttamente dall'interno delle vostre applicazioni, sia in modo sincrono che in modo asincrono.

Aspftp.zip

FTPServer
FTP, facilità di implementazione e sicurezza. Tutto questo è ciò che il presente pacchetto vi offre: troverete tutte le funzionalità necessarie per scrivere un server FTP completo.

Ftpserver.exe

PanelBarPro
Menu potenti ed attraenti per il vostro sito o per le vostre applicazioni web non costituiscono più un sogno.

Pbpro.msi

Sax.Net
La interconnessione tra varie applicazioni risulta essere facilitata da questo insieme di componenti. Ci aiuteranno a scrivere il codice ad hoc per realizzare le connessioni necessarie allo scambio dei dati.

Sax.net

Calcolare il codice fiscale

Si tratta di un WebServices reso pubblico dal sito *www.dotnethell.it*; i metodi esposti sono: *CodiceComune, ControllaCodiceFiscale e CalcolaCodiceFiscale*

Il framework .NET consente di invocare e utilizzare un WebServices in modo molto veloce e intuitivo; la quasi totalità delle operazioni, per integrare il componente all'interno della propria applicazione, viene svolta da un wizard dell'ambiente IDE di Visual Studio .NET.
All'utente basta quindi conoscere i soli metodi esposti dal WebService per invocarli, così come avviene per qualunque altro metodo o proprietà del linguaggio.
Per provare il WebService, senza "costruire" un'applicazione, è possibile visitare l'url *http://webservices.dotnethell.it/codicefiscale.asmx?op=CalcolaCodiceFiscale* e compilare in modo appropriato tutti i campi.
Nel CD-Rom allegato alla rivista e sul sito web *www.ioprogrammo.it* troverete una semplice applicazione Visual Basic .NET che richiama e utilizza il WebServices; in **Figura 1** è presente uno screenshot della stessa. Il codice che si occupa di invocare il WebServices è il seguente:

Fig. 1: ***Abbiamo realizzato una semplice applicazione che calcola il codice fiscale***

```
Dim cf As New it.dotnethell.
                        webservices.CodiceFiscale
Dim CodiceFiscale as String
CodiceFiscale= cf.CalcolaCodiceFiscale(
        "Gianfranco", "Forlino", "Cosenza",
                        "04/07/1976", "M")
```

Nel codice (sintassi VB.NET) viene dichiarato l'oggetto *cf* appartenente al WebServices e viene invocato il metodo *CalcolaCodiceFiscale* passandogli i parametri del caso, quest'ultimi "suggeriti" dall'IDE Visual Studio in fase di editing.

☑ I metodi esposti dal WebServices

CODICECOMUNE
Restituisce il codice del comune di nascita, passando il nome

CONTROLLACODICEFISCALE
Controlla la validità di un Codice Fiscale

CALCOLACODICEFISCALE
Effettua il calcolo del codice fiscale, passando come parametri, nome, cognome, comune di nascita, data di nascita in formato (dd/mm/aaaa) e sesso (M/F)

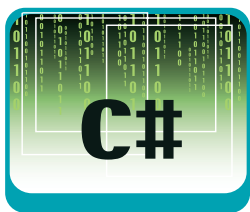
UTILIZZARE UN WEBSERVICE IN VISUAL STUDIO .NET

1 **Il primo passo per utilizzare il WebServices consiste nell'aggiungere al sistema i riferimenti per "rintracciarlo" in Rete. Dal menu *Progetto* dell'ambiente Visual Studio .NET scegliere la voce *Aggiungi Riferimento Web*, quindi, digitare l'URL http://webservices.dotnethell.it/codicefiscale.asmx?op=CalcolaCodiceFiscale nell'apposito campo.**

2 **Il sistema ricercherà il WebServices indicato e genererà, automaticamente (premere il pulsante *Aggiungere riferimento*), il codice per la sua fruizione. A Questo punto è possibile adoperare i metodi del WebService così come avviene per ogni altro oggetto del linguaggio.**

Grandi risultati con poche righe di codice

Conversioni di Valuta

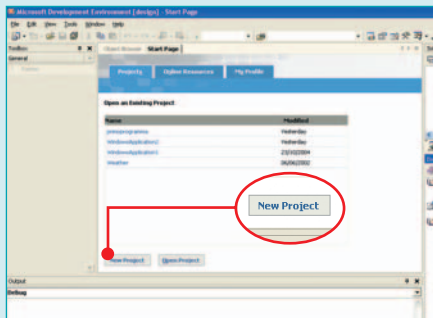
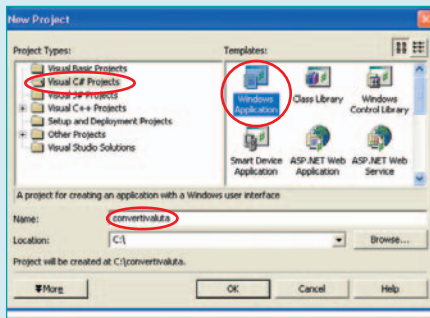
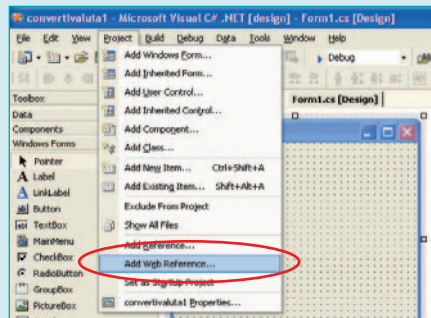
In questo articolo sfrutteremo un Web Service gratuito per creare in C# un'applicazione che converta una somma da una valuta all'altra al tasso di cambio attuale

Utilizza questo spazio per le tue annotazioni

L'applicazione che andremo a realizzare è di una semplicità disarmante grazie all'uso di un Web Service che farà per noi la maggior parte del lavoro sporco. Interrogheremo un Web Service esposto da *Xmethods* all'indirizzo *http://www.xmethods.net/sd/2001/CurrencyExchangeService .wsdl*. Il Web Service in questione espone un solo metodo: *getRate* che riceve in input due stringhe contenenti rispettivamente la denominazione del paese di partenza e la denominazione del paese d'arrivo. Il metodo *getRate* restituisce un valore *float* corrispondente al fattore di conversione dalla valuta di partenza e quella di destinazione. In pratica, al tasso corrente, passando *getRate* due stringhe contenenti rispettivamente "*Euro*" e "*Usa*" otterremo come valore di ritorno 1,2874 che è il corrispondente di un Euro in Dollari al tasso attuale.
Infine, ci complicheremo un po' la vita moltiplicando il valore restituito per un coefficiente che rappresenta la somma nella valuta di partenza. Ad esempio, per calcolare il corrispondente di 4 euro in dollari, moltiplicheremo per 4 il tasso di conversione ottenuto.

DALLA TEORIA ALLA PRATICA

Tralasciando le varie etichette e i contenuti grafici, la nostra form sarà così composta

1. Una casella di testo che chiameremo *"vlpartenza"*
2. Un comboBox che chiameremo *"cmpartenza"*
3. Un secondo comboBox che chiameremo *"cmdestinazione"*
4. Un bottone che ci servirà per avviare la conversione
5. Una label che chiameremo *"out1"* che ci servirà per contenere il valore convertito

IL CODICE ILLUSTRATO

1 **Avviamo un nuovo progetto cliccando su *"New Project"* nella *"start Page"*.**

2 **Il nuovo progetto si chiamerà *"convertivaluta"*. Sarà un progetto C# e una Windows Application.**

3 **Aggiungiamo il riferimento al *"web services"* utilizzando il menu *"Project/Add Web Reference"*.**

L'utente scriverà in *"vlpartenza"* le unità da convertire, in cmpartenza la valuta di partenza.
Ad esempio *vlpartenza=3 cmpartenza=euro.* In *cmdestinazione* l'utente inserirà invece il simbolo della valuta in cui si vuole ottenere la conversione ad esempio *cmdestinazione=USA*
Con questi valori, convertiremo 3 euro nel corrispettivo in dollari.

PICCOLE MIGLIORIE

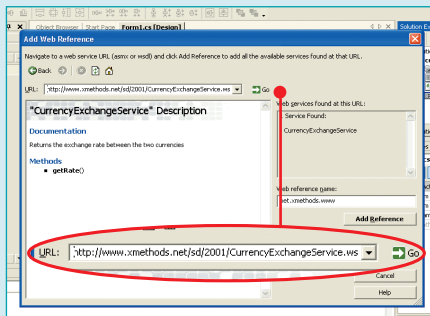
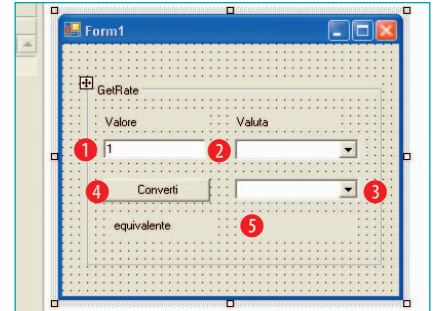
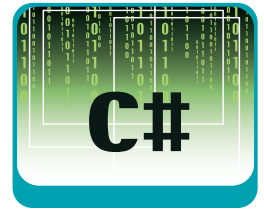
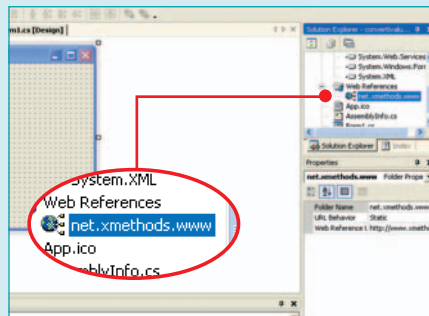
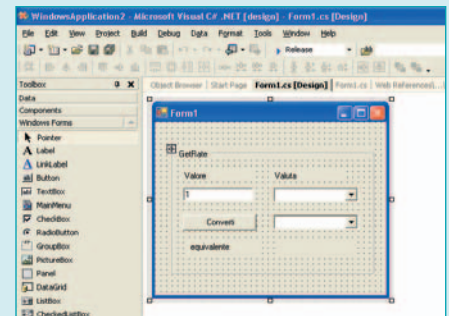
Abbiamo scelto di usare i combobox per evitare all'utente di dovere scrivere manualmente il codice della valuta. Nella lista prodotta dai combobox comparirà già un elenco di valute da cui scegliere. L'elenco dei codici delle valute è disponibile all'indirizzo *http://www.xmethods.net*. Vi si accede scendendo in basso nella pagina e utilizzando il link "*Curency Exchange Rate*". Nella pagina risultante in basso compare un elenco delle valute utilizzabili. Per riempire i nostri combobox non faremo altro che utilizzare la proprietà "*Items*" del combobox e riempire la collection incollando la lista dal link segnalato.

IL CODICE

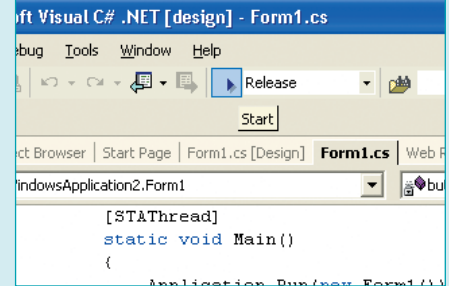
Non presenta difficoltà eccessive. Tutto viene gestito dall'evento *click* del bottone. Vengono dichiarate due variabili di tipo *float,* rispettivamente risultato e moltiplicatore. La variabile *risultato* viene riempita richiamando il Web Service in questione che prenderà in input *cmpartenza. SelectedItem.ToString()* e *cmdestinazione. SelectedItem.ToString()* e restituirà appunto un float contenente il tasso di scambio dell'unità di valuta.
Il passo successivo sarà moltiplicare il tasso di cambio ottenuto con il numero di unità dichiarare in *vlpartenza.Text*.

Le difficoltà sono praticamente nulle. Le uniche attenzioni sono relative al recuperare i codici di valuta utilizzando il metodo *SelectedItem* e passarli sotto la forma di stringa al Web Service utilizzando *ToString()*. Allo stesso modo, è importante convertire la *moltiplicatore.Text* in un tipo *single* utilizzando *moltiplicatore = Convert.ToSingle(vlpartenza.Text).*
Infine le due righe che instanziano il Web Service sono:

```
net.xmethods.www.CurrencyExchangeService ws=null;
ws =new net.xmethods.www.CurrencyExchangeService();
```

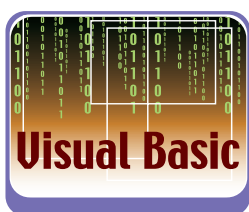
Semplice..

4 **Nella casella URL inseriamo il riferimento al web services, clicchiamo su *"go"* e infine su *"add reference"*.**

5 **Se tutto è andato a buon fine troveremo un riferimento al web services nella finestra *"solution explorer"*.**

6 **Riempiamo la form con i controlli necessari come descritto nell'articolo, trascinandoli dalla ToolBox presente sulla sinistra.**

```
private void button1_Click_1(object sender, System.EventArgs e)
{
    float risultato;
    float moltiplicatore=1;
    net.xmethods.www.CurrencyExchangeService ws=null;
    ws =new net.xmethods.www.CurrencyExchangeService();
    risultato = ws.getRate(cmpartenza.SelectedItem.ToString(), cmdestinazione.
                                                    SelectedItem.ToString());
    moltiplicatore = Convert.ToSingle(vlpartenza.Text);
    out1.Text = Convert.ToString(risultato*moltiplicatore);
}
```

7 **Clicchiamo due volte sul bottone e aggiungiamo il codice all'evento *OnClick*: il codice dell'applicazione è tutto qui!**

8 **Avviamo l'applicazione dalla ToolBar. Deve essere selezionata l'opzione *release*.**

Avicap e VB insieme per gestire una Webcam

La webcam che riconosce il volto

In questo articolo ci connettiamo a una videocamera, visualizziamo un filmato, salviamo frame e riprese su disco. Infine riconosciamo un volto all'interno di un'immagine

L'*AVIcap window class (avicap32.dll)* è una libreria per la gestione delle funzioni multimediali messe a disposizione da Windows. Consente di interfacciarsi a dispositivi di acquisizione audio/video senza doverne conoscere i dettagli hardware. Questo ci svincola dal dover programmare direttamente una determinata periferica e ci consente di poterci riferire a un unico insieme di comandi per pilotare qualunque dispositivo.
Ci occuperemo, prima di tutto, di realizzare una finestra attraverso la quale si abbia la possibilità di vedere quanto "catturato" dal device, la overlay window. Per realizzare questa finestra, i passi da compiere sono essenzialmente due:

- **Richiesta**: in questa prima fase viene chiesto al sistema di "destinare" una porzione della memoria video per consentire al proprio applicativo di "leggere" quanto ritornato dal device audiovisivo, specificandogli diverse informazioni come posizione e dimensioni.
- **Connessione:** in questa fase avviene la vera e propria connessione tra flusso video digitalizzato e la superficie di visualizzazione/cattura audiovisiva.

LE FUNZIONI DI BASE

Per realizzare i nostri scopi ci serviremo di due funzione definite all'interno della libreria *avicap32.dll: capGetDriverDescription*() e *capCreateCaptureWindow()*.
capGetDriverDescription() ci consente di ottenere informazioni circa i driver installati all'interno del sistema. *capCreateCaptureWindow()*, invece, ci consente di definire un'overlay window, realizzando la prima delle due fasi viste in precedenza.
Se la chiamata alla funzione *capCreateCaptureWindow()* va a buon fine, il valore restituito è un Long che rappresenta l'handle ad una nuova finestra, ossia quel dato che ci consentirà di passare alla fase successiva, quella di connessione.

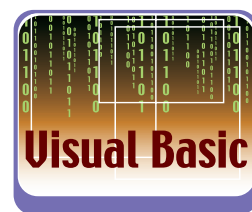
LE STRUTTURE PER CONTENERE I DATI

Avremo bisogno di tre strutture, che utilizzeremo per immagazzinare dati che ci serviranno nel corso dell'applicazione. Tutte le strutture sono definite in un modulo chiamato *VidCapFunctions.bas*. Nelle righe che seguono ho inserito un estratto delle dichiarazioni. Consiglio di aprire il modulo *VidCapFunctions.bas* dal file contenuto nel CD e visualizzare la definizione completa.

CAPSTATUS riporterà una serie di informazioni riguardanti lo stato attuale della finestra di capture, ad esempio, le dimensioni.

```
Type CAPSTATUS
    uiImageWidth As Long
    uiImageHeight As Long
[...]
    wNumVideoAllocated As Long
    wNumAudioAllocated As Long
End Type
```

La seconda struttura è quella che definisce le capacità del driver di cattura ed è denominata **CAPDRIVERCAPS**. La dichiarazione, all'interno di Visual Basic, è la seguente:

```
Type CAPDRIVERCAPS
    wDeviceIndex As Long
    fHasOverlay As
[...]
```

```
    hVideoExtIn As Long
    hVideoExtOut As Long
End Type
```

Se escludiamo il primo parametro, che definisce l'indice che identifica il driver che stiamo correntemente trattando), e gli ultimi quattro item della struttura, i restanti rappresentano, semplicemente, dei flag che illustrano alcune proprietà supportate dal driver in uso. In particolare, i tre flag *fHasDlgVideoSource*, *fHasDlgVideoFormat* e *fHasDlgVideoDisplay* ci consentono di stabilire se il driver supporta le dialog window *Video Source, Video Format* e *Video Display* che serviranno a richiamare le finestre utili alla modifica delle impostazioni di cattura video.

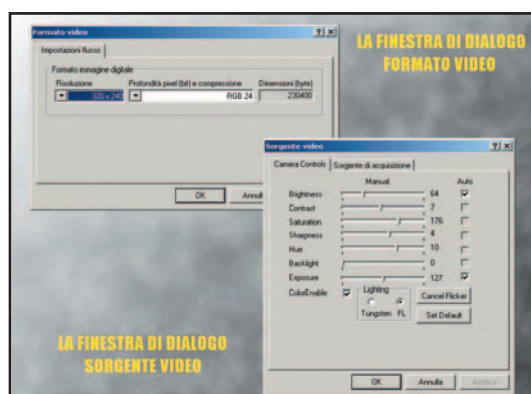



Fig. 1: ***Le finestre di dialogo Sorgente Video e Formato Video***

La terza ed ultima struttura sfruttata all'interno del programma è denominata **CAPTUREPARMS** e risulta così dichiarata:

```
Type CAPTUREPARMS
    dwRequestMicroSecPerFrame As Long
    fMakeUserHitOKToCapture As Long
[...]
    fDisableWriteCache As Long
    AVStreamMaster As Long
End Type
```

Contiene i parametri che controllano il processo di cattura dello streaming video.

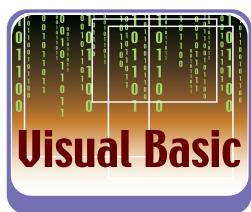
IL PROGETTO IN VISUAL BASIC

Il progetto, realizzato in Visual Basic 6, può essere suddiviso, praticamente, in due parti. La prima, si occupa d'implementare alcune tra le più comuni operazioni che solitamente si desiderano effettuare con un webcam: connessione, visualizzazione delle immagini "ritornate" dal device, modifica dei parametri di acquisizione, salvataggio di una sequenza video o, più semplicemente, di un singolo frame. La seconda parte, invece, si occupa di riconoscere, all'interno di un'immagine precedentemente salvata, l'area che racchiude il volto di una persona, contrassegnandola con un rettangolo rosso. Come vedremo più avanti, per raggiungere quest'ultimo scopo, faremo affidamento ad una libreria "recuperata" su Internet ed integrata opportunamente all'interno del programma.

L'INTERFACCIA

Iniziamo dalla prima parte del progetto, costituita da una sola form e da due moduli all'interno dei quali sono definite le funzioni utilizzate dal programma. In particolare:

- **frmPrincipale**: costituisce l'interfaccia principale del programma. Al suo interno sono presenti tutti i controlli necessari a visualizzare le immagini "ritornate" dalla propria webcam nonchè a consentirne la modifica delle impostazioni ed il salvataggio di un singolo frame o di una sequenza video;
- **frmGrabPreview:** rappresenta semplicemente una piccola finestra all'interno della quale è possibile visualizzare un'anteprima (in formato ridotto) del frame appena salvato;
- **Generale.bas:** questo modulo, com'è facile intuire, racchiude in sé tutte le dichiarazioni funzioni pubbliche e di uso generale (come la *SendMessage()*) utilizzate all'interno del progetto. Al suo interno, inoltre, sono definite diverse procedure che, sfruttando AVICAP32, consentono di compiere tutte le operazioni essenziali all'utilizzo della nostra webcam;
- **VidCapFunctions:** come si intuisce dallo stesso nome, al suo interno sono dichiarate tutte le costanti, le strutture e quant'altro necessario all'utilizzo della libreria AVICAP32. L'elenco delle strutture e soprattutto dei messaggi (costanti) dichiarati al suo interno, non è esaustivo e non tutti sono stati realmente sfruttati all'interno del progetto. La loro presenza è giustificata solo dalla volontà di "aggiungere" qualche informazione in più per consentire di migliorare il progetto stesso.

Fig. 2: ***L'interfaccia principale del programma***

COSA ACCADE ALL'AVVIO

Cominciamo a descrivere l'intero programma, partendo dall'avvio. Il primo evento innescato è, ovviamente, l'evento *Load()* della form principale:

```
Private Sub Form_Load()
    CaricaListaDevice
    With frmPrincipale
    If .cmbListaDevice="<Nessun device>" Then
        .Comando(0).Enabled=False
    End If
  End With
End Sub
```

Il suo compito primario è semplicemente quello di rilevare la lista dei device disponibili , driver di cattura, e caricarla all'interno del controllo ComboBox *cmbListaDevice* servendosi della procedura *CaricaListaDevice()* così definita:

```
Public Sub CaricaListaDevice()
  Dim strName As String
  Dim strVer As String
  Dim Ret As Boolean
  Dim Cont As Long
  Cont=0
  strName=Space(100)
  strVer=Space(100)
    Do
      'Preleva nome del driver e versione
        Ret=capGetDriverDescriptionA(
                                    Cont,strName,100,strVer,100)
      'Aggiungi le informazioni sul device trovato alla lista
        If Ret Then frmPrincipale.cmbListaDevice
                                        .AddItem Trim$(strName)
      Cont=Cont+1
    Loop Until Ret=False
  'Se ne esiste almeno uno...
    If Cont>1 Then frmPrincipale.cmbListaDevice
                                                  .ListIndex=0
End Sub
```

La procedura si serve della prima API menzionata ed appartenente alla libreria *avicap32.dll,* ossia la funzione *capGetDriverDescription()*. Al termine, se esiste almeno un device, lo seleziona, restituendo il controllo all'utente.

LE FUNZIONI DEI PULSANTI

La finestra *frmPrincipale* è costituita da diversi controlli di tipo *CommandButton*. Il primo pulsante che l'utente deve premere per poter iniziare a lavorare con la propria webcam è quello "etichettato" come *START*. Ovviamente, affinché tutto funzioni correttamente, è necessario che il dispositivo sia installato e risulti correttamente funzionante sul PC che si sta utilizzando. Il pulsante *START* è il primo elemento di un array di controlli di tipo *CommandButton* denominato *Comando*. La sua pressione dà luogo ad una serie di azioni, così come evidenziato dalle prime righe di codice inserite all'interno dell'evento *Click():*

```
Private Sub Comando_Click(Index As Integer)
  Dim Ret As Long
  Dim RetB As Boolean
  Select Case Index
        Case 0
            'Avvia se la label è Start...
              If Comando(0).Tag="Start" Then
                Avvia
                Comando(0).Picture=imgStop.Picture
                Comando(0).Tag="Stop"
              Else
                DisconnectAll
                Comando(0).Picture=imgStart.Picture
                Comando(0).Tag="Start"
              End If
...
```

L'evento associato al bottone è abbastanza semplice. Controlla la sua label. Se la label è impostata a *'Start'* avvia la connessione richiamando la procedura *avvia()* , cambia l'immagine di sfondo del bottone e imposta il valore della label a *'stop'*. Viceversa se la label è impostata a 'stop' effettua la disconnessione richiamando *DisconnectAll()*, cambia l'immagine di sfondo e setta la label a *'Start'*. Sia *avvia()* che *DisconnectAll()*, sono definite in *Generale.bas.*

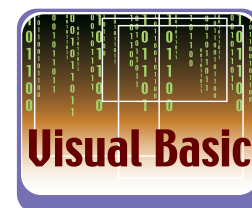
CONNESSIONE E DISCONNESSIONE

La procedura *Avvia()* si occupa di effettuare tutte le operazioni necessarie affinché ci si possa connettere al device selezionato. Essa risulta essere così costituita:

SPIEGHIAMOCI MEGLIO

La prima operazione importante che la procedura compie è quella di creare la finestra che ospiterà le riprese della webcam. Per far questo, la procedura si affida alla funzione *capCreateCaptureWindow()*, passandole i parametri essenziali all'operazione. In

CREIAMO LA FINESTRA

```
Sub Avvia()
  Dim WindowName As String
  Dim dwStyle As Long
  Dim x As Long, y As Long
```

```
Dim nWidth As Long, nHeight As Long
Dim hParentWin As Long
Dim Ret As Long
Dim RetB As Boolean
'Imposta le caratteristiche della capture window
  WindowName="My Capture Win"
  dwStyle=WS_CHILD+WS_VISIBLE
  nWidth=320
  nHeight=240
  hParentWin=frmPrincipale.Picture1.hwnd
  x=y=0
'Creazione di una Capture Window
hWndCapture=capCreateCaptureWindow(
      WindowName,dwStyle,x,y,nWidth,nHeight,
                                   hParentWin,0)
'Callback...
  capSetCallbackOnStatus hWndCapture,
                       AddressOf NewStatusCallback
'Connessione della Capture Window
'al capture driver
Ret=SendMessage(hWndCapture,
          WM_CAP_DRIVER_CONNECT,iDevice,0)
```

1 **Imposta la Overlay Windows come una finestra di 320x240. Prende l'handle della finestra e lo collega al driver della periferica tramite la *SendMessage*.**

CONNETTIAMOCI AL DRIVER

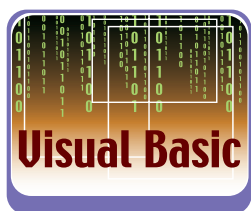
```
If Ret Then
   'Richiesta delle capacità del capture driver
   RetB=SendMessage(
    hWndCapture,WM_CAP_DRIVER_GET_CAPS,
           Len(S_CapDrvCaps), S_CapDrvCaps)
   'Richiesta stato attuale capture window
   RetB=SendMessage(
         hWndCapture,WM_CAP_GET_STATUS,
                   Len(S_CapStat),S_CapStat)
   'Richiesta capacità hardware del driver
   RetB=SendMessage(hWndCapture,
            WM_CAP_GET_SEQUENCE_SETUP,
         Len(S_CapParms),  S_CapParms)
   'Abilita tutti i comandi
   AbilitaComandi (True)
   'Disabilita gli eventuali dialog non supportati
   If S_CapDrvCaps.fHasDlgVideoFormat=0
Then frmPrincipale.Comando(1).Enabled=False
   If S_CapDrvCaps.fHasDlgVideoDisplay=0
Then frmPrincipale.Comando(2).Enabled=False
   If S_CapDrvCaps.fHasDlgVideoSource=0
Then frmPrincipale.Comando(3).Enabled=False
   'Se è possibile il modo overlay, allora abilitalo,
   'in caso contrario imposta il Preview rate
   If S_CapDrvCaps.fHasOverlay Then
      Ret=SendMessage(hWndCapture,
                 WM_CAP_SET_OVERLAY, 1, 0)
   Else
      Ret=SendMessage(hWndCapture,
          WM_CAP_SET_PREVIEWRATE, 120, 0)
      Ret=SendMessage(hWndCapture,
```

```
                       WM_CAP_SET_PREVIEW, 1, 0)
   End If
      WindowResize
   Else
      Call MsgBox("Non è stato possibile
   connettersi ad alcun device"&vbCrLf, vbCritical
                          + vbOKOnly, "Errore!")
      Unload frmPrincipale
   End If
   'Imposta le informazioni sui flag
     MostraFlag
End Sub
```

2 **Tenta una connessione al driver della periferica. Se va a buon fine controlla le capacità del driver ed imposta eventualmente le finestre di dialog e un eventuale overlay.**

particolare, imposta il nome della capture window secondo una stringa "personale" e definisce, come parent window (ossia la finestra che "ospiterà" l'overlay window), il controllo *Picture1* posto all'interno della form principale. Inoltre, cosa importante, determina lo stile della finestra come *WS_CHILD* in modo tale da essere "certi" che risulti realmente come una finestra figlia di *Picture1*. In caso contrario, ossia se omettessimo di scrivere il flag *WS_CHILD* nella definizione del parametro *dwStyle*, avremmo una flat window separata dalla form principale. Se tutto è stato eseguito correttamente, la funzione *capCreateCaptureWindow()* ritorna l'handle alla capture window che servirà, d'ora in poi, come riferimento univoco per le successive operazioni. Tralasciamo, per il momento, la successiva istruzione, quella etichettata come *"Callback…"* perché sarà oggetto di discussione in seguito e passiamo oltre. Nel successivo passo, dopo che abbiamo ottenuto l'handle alla finestra di cattura video, possiamo collegarla al dispositivo mediante l'invio di un opportuno messaggio definito dalla costante *WM_CAP_DRIVER_CONNECT* e destinato proprio a questa finestra. Inutile sottolineare che ci serviremo dell'ormai nota funzione *SendMessage()* per raggiungere lo scopo.
Da questo momento in poi, siamo in grado di recuperare tutte le informazioni che ci servono per gestire la nostra webcam, oltre, chiaramente, a poter modificare i suoi parametri accedendo alle finestre di dialogo supportate dal dispositivo e viste precedentemente. Innanzitutto, recuperiamo le informazioni che ci servono, attraverso l'invio di tre messaggi opportuni che "riempiranno" tre strutture del tipo descritto all'inizio. In particolare, dopo aver ottenuto le informazioni sulle capacità del driver video, riposte all'interno della struttura *S_CapDrvCaps*, controlliamo i flag relativi al supporto delle varie dialog window e disabilitiamo, laddove necessario, i pulsanti analoghi riposti sulla form *frmPrincipale*.
La procedura *Avvia()*, come ultima istruzione, mo-

NOTA

Maggiori dettagli relativi all'argomento possono essere recuperati direttamente dall'MSDN al link:
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/multimed/htm/_win32_about_video_capture.asp

stra i valori dei flag di ciascuna struttura all'interno di opportune etichette disposte nella parte inferiore della form.

DISCONNESSIONE

La disconnessione dal dispositivo rappresenta senza dubbio un'operazione di gran lunga più semplice della precedente. Infatti, la procedura *DisconnectAll()*, richiamata proprio per raggiungere quest'obiettivo, è così definita:

```
Function DisconnectAll()
  Dim Ret As Long
  Ret=SendMessage(hWndCapture,
                              WM_CAP_SET_PREVIEW,0,0)
  'Interrompe il preview
  Ret=SendMessage(hWndCapture,
               WM_CAP_DRIVER_DISCONNECT,iDevice,0)
  'Disconnette il capture driver
  Ret=DestroyWindow(hWndCapture)
  'Distrugge la finestra
  AbilitaComandi (False)
End Function
```

NOTA

OVERLAY O NON OVERLAY?

È bene osservare che buon esito della procedura di connessione, non è vincolato solo alla corretta chiamata della funzione *SendMessage()*, ma dipende anche da altri fattori, meglio definiti come *capacità hardware del dispositivo*. Queste informazioni sono restituite all'interno di una struttura di tipo CAPDRIVERCAPS, recuperate attraverso l'istruzione:

```
RetB=SendMessage(hWndCapture,WM_CAP_DRIVER_GET_CAPS,Len(
                                        S_CapDrvCaps),S_CapDrvCaps)
```

In particolare, la procedura *Avvia()* analizza il flag *fHasOverlay* che identifica la capacità o meno del driver di generare un overlay hardware. Un valore di *fHasOverlay* pari ad 1 indica che, connettendo il driver di cattura ad una superficie di visualizzazione e cattura, potremo disporre di una connessione diretta,ossia senza alcun intervento da parte del processore, fra segnale di ingresso digitalizzato e memoria video. In questo caso si parla, per l'appunto, di *overlay hardware*.
In caso contrario, ossia se *fHasOverlay* vale 0, ciò indica l'indisponibilità da parte del dispositivo d'acquisizione di generare un overlay hardware, con ovvio "intervento" da parte della CPU che si dovrà preoccupare della lettura del segnale digitalizzato e della sua scrittura all'interno di una porzione di memoria video. Questa seconda procedura prende il nome di *Preview* e presenta maggiori limitazioni sia dal punto di vista della qualità dell'immagine che della velocità d'acquisizione.
Tutto questo discorso, quindi, giustifica le seguenti righe di codice:

```
'Se è possibile il modo overlay, allora abilitalo,
        'in caso contrario imposta il Preview rate
          If S_CapDrvCaps.fHasOverlay Then
             ret=SendMessage(hWndCapture,WM_CAP_SET_OVERLAY,1,0)
          Else
             ret=SendMessage(hWndCapture,WM_CAP_SET_PREVIEWRATE,120,0)
             ret=SendMessage(hWndCapture,WM_CAP_SET_PREVIEW,1,0)
          End If
...
```

A questo proposito non c'è molto da aggiungere vista la semplicità delle operazioni eseguite e possiamo tranquillamente passare ad illustrare come gestire ora il nostro device.

CONNESSI? ORA DIVERTIAMOCI!

Al termine della procedura di avvio e collegamento al dispositivo, siamo finalmente pronti per poter "giocare" con la nostra webcam. Se tutto è andato per il verso giusto, la webcam dovrebbe "proiettare" le immagini catturate all'interno della finestra disposta sulla form principale. Da questo momento abbiamo a disposizione diverse opportunità. Sulla sinistra della form, infatti, sono disposti sei pulsanti. I primi quattro si occupano di richiamare le tre finestre di dialogo viste all'inizio dell'articolo e quella riguardante la gestione della compressione video. Come spiegato a proposito del pulsante *START*, anche questi controlli fanno parte dell'array di *CommandButton* denominato *Comando* e gestiti dall'evento *Click()* comune. In particolare, per gli item con indice 1, 2, 3 e 4 (che si occupano di richiamare le finestre di dialogo menzionate), abbiamo:

```
...
  Case 1
    'Richiama la finestra di dialogo Formato Video
    Ret=SendMessage(
       hWndCapture,WM_CAP_DLG_VIDEOFORMAT,0,0)
    WindowResize
  Case 2
    'Richiama la finestra di dialogo Display Video
    Ret=SendMessage(
          hCapWnd,WM_CAP_DLG_VIDEODISPLAY,0,0)
  Case 3
     'Richiama la finestra di dialogo Sorgente Video
     Ret=SendMessage(
      hWndCapture,WM_CAP_DLG_VIDEOSOURCE,0,0)
  Case 4
     'Richiama la finestra di dialogo Compressione Video
     Ret=SendMessage(hWndCapture,
            WM_CAP_DLG_VIDEOCOMPRESSION,0,0)
...
```

Come si può vedere, non è molto complicato ottenere il risultato sperato. Le uniche cose importanti da tenere a mente sono l'handle della finestra di cattura video e la corretta costante che definisce il messaggio che *SendMessage()* le dovrà inviare. Si osservi sin da ora l'intenso uso e l'importanza che la funzione *SendMessage()* assume all'interno dell'intero progetto. I restanti due pulsanti presenti sulla sinistra del form si occupano di effettuare il salvataggio di un singolo frame (*grabbing*) o di una sequenza video (*streaming*) all'interno di un opportuno file. Anche

in questo caso, il codice che definisce le azioni da intraprendere successivamente alla pressione di uno di questi due pulsanti, è contenuto all'interno dell'evento *Click()* dell'array di controlli *Comando*.

SALVARE UN'IMMAGINE SU DISCO

La procedura che consente di salvare in un file un singolo frame è piuttosto banale. Esiste più di una possibilità per raggiungere lo scopo, ma la via più semplice è quella mostrata all'interno del progetto.

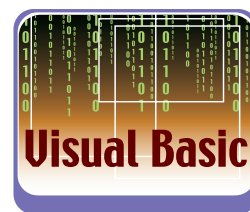
```
...
Case 5
   'Consente il salvataggio del frame corrente
   Ret=SendMessage(
      hWndCapture,WM_CAP_FILE_SAVEDIB,0&,ByVal
                                txtNomeFrame.Text)
   If Ret=0 Then
      MsgBox "Salvataggio frame non riuscito!",
                                vbCritical,"Errore!"
   Else
      MsgBox "Salvataggio frame riuscito!"
      'Se l'utente ha selezionato di vedere l'anteprima,
      'allora mostrala
      If frmPrincipale.CheckGrabPreview.Value=1 Then
         frmGrabPreview.Show
         frmGrabPreview.Image1.Picture=
            LoadPicture(frmPrincipale.txtNomeFrame.Text)
      End If
   End If
Case 6
   'Consente il salvataggio della sequenza video
   SalvaVideo
```

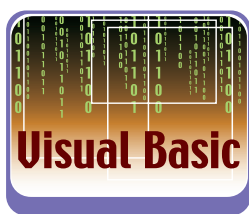
Tutto quello che bisogna fare è inviare all'overlay window un opportuno messaggio, definito dalla costante *WM_CAP_FILE_SAVEDIB*, specificando, come ultimo parametro della funzione *SendMessage()*, il nome da assegnare al file. Al termine di quest'operazione, se l'utente ha selezionato l'opzione di visualizzazione dell'anteprima (attraverso l'uso del controllo *CheckGrabPreview*), il programma mostra a video la finestra *frmGrabPreview* che riporta, in un formato più piccolo, il frame appena salvato. Esistono altre considerazioni riguardanti le operazioni di grabbing, ma per gli scopi dell'articolo sono ininfluenti e verranno quindi tralasciate.

SALVIAMO UNO SPEZZONE DI FILMATO

Il caso 6 della procedura precedente avvia la procedura *salvavideo*. L'operazione di streaming, è senza dubbio più complessa della precedente e necessita di maggiore attenzione. La procedura SalvaVideo() può essere la seguente

```
Public Sub SalvaVideo()
 Dim RetB As Boolean
 Dim VideoFileName As String
 'Nome del file che costituirà la sequenza
   VideoFileName=frmPrincipale.txtNomeFilmato.Text
 'Preleva i dati relativi alla struttura CAPTUREPARMS
   RetB=SendMessage(hWndCapture,
                          WM_CAP_GET_SEQUENCE_SETUP,
                  VarPtr(S_CapParms), Len(S_CapParms))
 'Imposta alcuni items secondo i desideri
 'Numero di frame per secondo (30)
     S_CapParms.dwRequestMicroSecPerFrame=
                                    (1*(10 ^ 6))/30
   'L'utente deve premere OK per iniziare
     S_CapParms.fMakeUserHitOKToCapture=True
   'Non catturiamo l'audio
     S_CapParms.fCaptureAudio=False
   'Impostiamo lo STOP con la pressione del pulsante
   'destro del mouse (oltre a quello predefinito ESC)
     S_CapParms.fAbortRightMouse=True
 'Salva le nuove informazioni
   RetB=SendMessage(hWndCapture,
                          WM_CAP_SET_SEQUENCE_SETUP,
          VarPtr(S_CapParms), ByVal (Len(S_CapParms)))
 'Inizia la registrazione video
   RetB=SendMessage(hWndCapture,
                             WM_CAP_SEQUENCE,0,0)
 'Salva il file CAPTURE.AVI in quello definito da
 'VideoFileName
   RetB=SendMessage(hWndCapture,
                  WM_CAP_FILE_SAVEAS,0,VideoFileName)
End Sub
```

La prima operazione importante è la definizione del nome del file AVI all'interno del quale salvare la sequenza video. La seconda è il recupero delle informazioni utili ad effettuare correttamente un'operazione di streaming. Quest'ultima operazione,viene realizzata sempre attraverso l'invio di un opportuno messaggio, *WM_CAP_GET_SEQUENCE_SETUP*, che non fa altro che valorizzare una struttura del tipo *CAPTUREPARMS* denominata *S_CapParms.* In realtà, questa azione non è realmente necessaria, ma può essere utile per leggere i parametri del dispositivo prima di avviare la suddetta operazione. Le istruzioni successive si occupano dell'impostazione di alcuni item della struttura prima menzionata e, successivamente, della memorizzazione di tali modifiche, ad esempio il numero di frame per second. In particolare, viene impostato l'item *fAbortRightMouse* che consente di terminare l'operazione di streaming attraverso la pressione del pulsante destro del mouse (oltre che del predefinito tasto *ESC*).
Solo ora, dopo queste considerazioni, possiamo av-

STREAMING
Con il termine *streaming* ci riferiamo al processo di registrazione di una sequenza video riprodotta nella finestra di overlay.

GRABBING
Con *grabbing* si intende il procedimento di cattura di un singolo fotogramma dal flusso video riprodotto all'interno della nostra finestra di overlay.

viare la registrazione, che verrà eseguita proprio attraverso le ultime due istruzioni.

FEEDBACK ALL'UTENTE

Effettuando alcune prove con la vostra webcam, noterete senza dubbio un particolare importante ossia lo scorrimento in tempo reale di alcune informazioni durante la fase di streaming.

Fig. 5: *I dettagli mostrati durante l'operazione di streaming*

Presenti all'interno della statusbar della form principale, queste informazioni sono ottenute in tempo reale attraverso la definizione di un'opportuna funzione di callback (un'operazione possibile solamente attraverso l'uso della parola chiave *AddressOf*) definita dall'istruzione nella procedura *Avvia()*.

```
capSetCallbackOnStatus hWndCapture,AddressOf
                                   NewStatusCallback
```

In particolare, la funzione *capSetCallbackOnStatus()* è così definita:

```
Public Function capSetCallbackOnStatus(ByVal
    hwndPIC As Long, ByVal lpProc As Long) As Boolean
  capSetCallbackOnStatus=SendMessageL(hwndPIC,
          WM_CAP_SET_CALLBACK_STATUS,0,lpProc)
End Function
```

capSetCallbackOnStatus() non fa altro che sostituire la procedura predefinita richiamata durante un'operazione di streaming con una procedura definita all'interno del nostro progetto. Per essere maggiormente precisi, ecco il codice che implementa la procedura *NewStatusCallback()*:

```
Public Function NewStatusCallback(ByVal hwnd As Long,
 ByVal iID As Long,ByVal PStatusText As Long) As Long
  Dim sStatusString As String
  Dim uStatusString As String
  Dim NumeroFotogrammi As Integer
  Dim Pos As Integer
  If iID=0 Then Exit Function
  'Conversione del puntatore ricevuto nella stringa
  'e conversione in formato Unicode
    sStatusString=String$(255,0)
    lStrCpy StrPtr(sStatusString),PStatusText
    uStatusString=StrConv(sStatusString,vbUnicode)
  'Mostra il messaggio nella StatusBar
    frmPrincipale.StatusBar1.SimpleText=uStatusString
      Select Case iID
        Case IDS_CAP_BEGIN:
          frmPrincipale.imgObiettivoOn.Visible=True
          frmPrincipale.imgObiettivoOn.Refresh
        Case IDS_CAP_END
          frmPrincipale.imgObiettivoOn.Visible=False
          frmPrincipale.imgObiettivoOn.Refresh
          frmPrincipale.ProgressBar1.Value=0
          frmPrincipale.lblFotogrammiAcquisiti.Caption=""
        Case IDS_CAP_STAT_VIDEOCURRENT
          Pos=InStr(1,uStatusString,"(")
          NumeroFotogrammi=CInt(Mid$(
                            uStatusString,23,Pos-23))
          'Estrai il numero di fotogrammi acquisiti
          'dal messaggio del tipo "Fotogrammi
          'acquisiti: 34 (scartati: 19) - 2:479 sec."
            If NumeroFotogrammi>0 Then
              frmPrincipale.ProgressBar1.
                       Value=NumeroFotogrammi
              frmPrincipale.lblFotogrammiAcquisiti.
                       Caption=NumeroFotogrammi
              DoEvents
            End If
        'Case ?
        'INSERIRE QUI L'EVENTUALE GESTIONE DI
                                   ALTRI MESSAGGI
  End Select
End Function
```

Ogni qualvolta la procedura viene richiamata a seguito di un certo evento (come l'inizio dello streaming), le vengono passati tre parametri tra cui l'ID del messaggio ed una stringa di testo che riporta diverse informazioni. Se si conosce il formato di ogni stringa correlata ad ogni possibile messaggio, è possibile estrarre alcune informazioni in tempo reale o, semplicemente, riportare la stringa passata

come parametro all'interno di una barra di stato (così come fatto nel nostro progetto). Per non appesantire troppo il codice, sono stati gestiti solo tre messaggi *IDS_CAP_BEGIN, IDS_CAP_END* e *IDS_CAP_STAT_VIDEOCURRENT,* lasciando ai più volenterosi il compito di migliorare la procedura. I primi due messaggi rappresentano sempre il primo e l'ultimo messaggio inviato alla funzione di callback rispettivamente nel momento in cui inizia e termina la registrazione.

RICONOSCIMENTO DEL VOLTO

A questo punto siamo arrivati all'ultima parte dell'intero progetto ossia l'implementazione di una procedura di face recognition. Attraverso questa funzionalità, il programma è in grado di delimitare con un rettangolo rosso l'area che racchiude il viso della persona raffigurata all'interno del singolo frame catturato. Le capacità "aggiunte" al nostro programma sono merito di una piccola libreria, prelevata gratuitamente su Internet ed allegata al presente progetto. Analogamente a quanto riportato nella prima parte, mostriamo subito le form ed i moduli associati a questa seconda parte del progetto:

- **frmDetectFace:** questa form consente di aprire un file d'immagine per determinare l'area all'interno della quale è contenuto il viso della persona fotografata.
- **Bufalo.bas:** contiene le dichiarazioni per l'algoritmo di riconoscimento. In particolare, al suo interno, sono dichiarate due funzioni appartenenti alla libreria *Bufalo.dll*, anch'essa gratuitamente prelevabile da Internet.

Quando l'utente preme il pulsante a destra della form principale ossia il controllo *cmdScanFace*, gli viene mostrata a video la form *frmDetectFace*.
All'interno di questa finestra sono contenuti tre controlli principali: due pulsanti ed una picturebox. La pressione del primo pulsante consente di selezionare il file bitmap da aprire per la scansione, mentre il secondo avvia la procedura di scansione vera e propria per determinare l'area che delimita il viso della persona. In particolare, la pressione di quest'ultimo pulsante, *cmdDetectFace*, innesca la seguente serie d'azioni:

```
Private Sub cmdDetectFace_Click()
 Dim ret As Long
 Dim xSrc As Long
 Dim ySrc As Long
 Dim nWidth As Long
 Dim nHeight As Long
 Dim mFace As FaceRect
 mFace.height=0
 mFace.width=0
 mFace.xSrc=0
 mFace.ySrc=0
 GetImage (mFilename)
    numfaces=DetectFaces(dbitmap.bmWidth, dbitmap
        .bmHeight, dbitmap.bmBitsPixel, dbitmap.bmBits)
    For n=0 To numfaces - 1
        ret=GetFaceRect(n,mFace)
        If ret=0 Then
        `linea superiore
          Me.Picture1.Line (mFace.xSrc, mFace.ySrc)-
         (mFace.xSrc+mFace.width, mFace.ySrc), vbRed,B
         `linea sinistra
          Me.Picture1.Line (mFace.xSrc, mFace.ySrc)-
         (mFace.xSrc, mFace.ySrc+mFace.height), vbRed,B
          `linea inferiore
          Me.Picture1.Line (mFace.xSrc, mFace.ySrc+
              mFace.height)-(mFace.xSrc+mFace.width,
                  mFace.ySrc+mFace.height), vbRed,B
          'linea destra
          Me.Picture1.Line (mFace.xSrc+mFace.width,
               mFace.ySrc)-(mFace.xSrc+mFace.width,
                  mFace.ySrc+mFace.height), vbRed,B
        End If
    Next n
    DeleteObject hbitmap
End Sub
```

Come potete notare, queste righe non fanno altro che "ciclare" sulle facce riconosciute, tracciando un rettangolo per ognuno di esse. Non credo sia molto complicato comprendere il listato precedente. L'unica cosa da aggiungere riguarda l'utilizzo delle due funzioni *DetectFaces()* e *GetFaceRect()* dichiarate all'interno del modulo *Bufalo.bas*
La prima funzione inizializza le strutture necessarie a determinare il numero di facce possibili. La seconda, invece, determina le coordinate relative ad ogni faccia rintracciata consentendo di disegnare il rettangolo che delimita quest'area.

Fig: 6: **Il volto è stato riconosciuto**

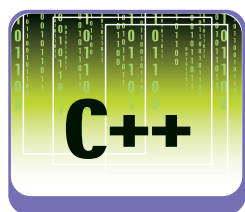
CONCLUSIONI

Resterebbe ovviamente ancora molto da dire (ad esempio, non si è accennato alle funzionalità audio) ma, come appena sottolineato, l'approfondimento di questo ed d'ulteriori aspetti, è lasciato alla personale curiosità di ognuno di voi.

Francesco Lippo

Un engine 3D per sviluppare rapidamente un videogioco

Un gioco 3D in venti righe

In questo articolo creeremo una stanza con vista tridimensionale, ci metteremo dentro una colonna, applicheremo le texture e infine accenderemo le luci!

Utilizza questo spazio per le tue annotazioni

Quando si parla di motori grafici tridimensionali in tanti pensano che la questione si possa ridurre a una più o meno lunga chiacchierata su Direct3D e/o OpenGL. Non è esattamente così.
Un'approssimazione più vicina alla realtà è quella di utilizzare dei motori per la gestione della grafica 3D. Questi motori interagiscono con Directx o OpenGL rendendo più agevole il compito del programmatore. In sostanza, un engine 3D offre al programmatore una serie di funzioni che gli consentono di non conoscere lo strato sottostante, ma piuttosto di usare una sorta di macro per inizializzare una scena o aggiungere una videocamera o gestire il movimento. In questo articolo mostreremo come si possa iniziare a creare un ambiente per la gestione di un gioco 3D utilizzando Irrlicht. Un game engine interessante perché utilizzabile con moltissimi linguaggi inclusi .NET e C++. Per questo articolo utilizzeremo C++ ma con pochi accorgimenti potrete usare irrlicht anche nei vostri progetti .NET.

SETUP DELLA LIBRERIA

Potete ovviamente utilizzare qualunque compilatore C++ per realizzare questo progetto. La procedura seguente è riferita al Microsoft Visual C++.
La prima cosa da fare, per utilizzare IrrLicht, è impostare opportunamente i file di inclusione e le librerie. Per fare ciò occorre selezionare dal menu principale *"Tools/Options/Directories"* e inserire tra le cartelle di ricerca quella contenente il file *"irrlicht.h"* (sezione *"Includes"*) e quella contenente il file *"irrlicht.lib"* (sezione *"Libraries"*).
Fatto questo è utile caricare uno dei progetti di esempio (reperibili nella sotto-cartella *"examples"*) e provare a compilarlo. Se tutto va per il verso giusto e non ci sono errori, il nostro IDE è correttamente configurato per l'utilizzo di IrrLicht.
Facendo uso di altri IDE la procedura è analoga: bisogna fare in modo che il compilatore "veda" i file di inclusione e le librerie.

NOTA

COME FACCIO A PROCURARMI QUESTA IRRLICHT?

IrrLicht è reperibile all'indirizzo http://irrlicht.sourceforge.net/index.html. Oppure, ovviamente la trovate all'interno del CD di ioProgrammo. La libreria viene aggiornata e migliorata spessissimo. Per l'esempio che vi proponiamo in questo articolo è sufficiente la versione che trovate nel nostro CD. Se decidete di usarla in progetti più complessi non dimenticate di controllare, di tanto in tanto, il rilascio di nuovi upgrade. In ogni caso ioProgrammo è sempre vigile, per cui troverete notizie sulle eventuali novità anche sulla nostra rivista

UNA STRUTTURA UNIVERSALE

Quella che presentiamo nell'esempio è la struttura base di un programma che utilizza IrrLicht.
Nel **passo 1** avviene il setup generale della libreria. Nell'ordine:

- viene incluso il file header *"irrlicht.h"*
- si dichiara che si utilizzeranno i namespace tipici di IrrLicht (*"irr"* è quello primario, seguito dai 5 specifici *"core"*, *"scene"*, *"video"*, *"io"* e *"gui"*)
- si dichiara, tramite la direttiva *"#pragma"* che si utilizzerà la libreria *"irrlicht.lib"*

A questo punto qualsiasi chiamata a una funzione

o oggetto di IrrLicht sarà riconosciuta e linkata correttamente. Questa inizializzazione va sempre eseguita.

Nel **passo 2** avviene la creazione del *"device"*, cioè l'oggetto col quale si creano e gestiscono tutti gli altri oggetti del motore. Come si vede viene creato un puntatore a un oggetto di tipo *IrrlichtDevice*, tramite la funzione *createDevice()*. Quest'ultima accetta 7 parametri. Il dettaglio lo trovate in **Tabella 1**.
Si può facilmente intuire l'importanza della funzione *createDevice()* in quanto è qui che vengono stabilite la maggior parte delle impostazioni fondamentali per la resa grafica del programma.

Nel **passo 3**, il dispositivo *"device"* appena creato viene utilizzato per stabilire il titolo della finestra, attraverso la funzione *setWindowCaption()*. Si noti come questa funzione accetti stringhe codificate con caratteri a 16 bit (unicode), da qui la presenza della lettera *"L"* prima della stringa del titolo stessa. Inutile dire che l'utilizzo di caratteri a 16 bit garantisce estrema libertà in fase di localizzazione del software: si pensi ad esempio all'utilizzo del tutto indolore dei caratteri asiatici. Sempre utilizzando *"device"* si ottengono i puntatori al driver video (*IVideoDriver* driver*) e al gestore della scena 3D (*ISceneManager* smgr*) che saranno utilizzati inseguito per il disegno a schermo della scena in memoria.

Nel **passo 4**, vi è la funzione che si occupa di caricare l'oggetto 3D in memoria. È sufficiente specificare il nome del file ("*colonna.x*"), senza preoccuparsi minimamente della sua struttura interna: a questo pensa IrrLicht.

Notare l'utilizzo di un puntatore a oggetto di tipo *IAnimatedMesh*: sebbene la nostra colonna non abbia alcuna animazione, questo ci permetterà di estendere il programma in seguito senza modificarne il codice. Per adesso consideriamo il file in esame come un oggetto con un solo frame di animazione. Una volta creata la mesh che desideriamo, è necessario aggiungerla alla scena da disegnare: questo viene fatto creando un "nodo" nella struttura ad albero della scena, che contenga la mesh. Si utilizza la funzione *addAnimatedMeshSceneNode()* che restituisce il puntatore al nodo, il quale viene salvato nella variabile *"node"*. È inoltre creata una videocamera che riprenderà l'intera scena tramite la funzione *addCameraSceneNode()* nella quale si specifica la posizione della videocamera e la direzione in cui sta "guardando" (in altre parole, si crea un vettore il cui punto di applicazione è la posizione desiderata).

Il **passo 5** costituisce il ciclo fondamentale dell'applicazione. La condizione di uscita da questo ciclo è

IL CODICE ILLUSTRATO

```
#include <irrlicht.h>
using namespace irr;
using namespace core;
using namespace scene;
using namespace video;
using namespace io;
using namespace gui;
#pragma comment(lib, "Irrlicht.lib")
```

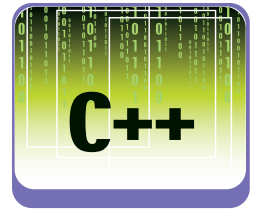
1 PREPARIAMO GLI STRUMENTI
8 righe per includere la libreria "Irrlicht", dichiarare i namespace, avvertire il compilatore che useremo la libreria *irlicht*.

```
int main() {
IrrlichtDevice *device =
                    createDevice(EDT_DIRECTX8,
              dimension2d<s32>(640, 480), 16,
                          false, false, false, 0);
```

2 APRIAMO UNA FINESTRA
2 righe per dire che il nostro gioco sarà visualizzato in una finestra di 640x480 pixel e avrà 16 bit per colore

```
device->setWindowCaption(L"ioProgrammo
                              con IrrLicht");
IVideoDriver* driver = device->getVideoDriver();
ISceneManager* smgr =
                     device->getSceneManager();
```

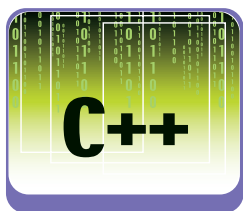
3 METTIAMO UN TITOLO ALLA FINESTRA
3 righe per definire il titolo della finestra, inizializzare un puntatore al gestore del video e uno al gestore della scena

posta sul *"device"* creato all'inizio: in pratica si dice al software di continuare a funzionare fin quando è attivo il device. Questo non è più vero quando, ad esempio, si chiude la finestra dell'applicazione o si preme *ALT+F4* sotto Windows. Tutte le operazioni di disegno (sia quelle 3D che quelle 2D o anche quelle

CHE COS'È UN FILE .X ?
I file *.x* sono quelli specifici delle DirectX, ma è possibile caricare allo stesso modo anche file *.3ds* (3D Studio Max), *.obj* (Maya), *.bsp* (livelli di Quake3) e molti altri.

UNA LIBRERIA UNIVERSALE
IrrLicht ha, tra i suoi molteplici pregi, quello di potere utilizzare come librerie grafiche sia le Direct3D che le OpenGL. Il funzionamento è garantito su diverse piattaforme tra le quali figurano tutti i Windows dal 98 a XP e Linux. Il supporto per MacOS è pianificato ma tuttora ancora non implementato.

I PARAMETRI ACCETTATI DA CREATEDEVICE

deviceType	è il tipo di "device" (= dispositivo) utilizzato per la gestione della grafica. Può essere un device software (EDT_SOFTWARE), Direct3D (EDT_DIRECTX8, EDT_DIRECTX9) o OpenGL (EDT_OPENGL)
windowSize	le dimensioni della finestra (o dello schermo se in modalità full-screen)
bits	la profondità di colore in bit. Può essere 16 o 32.
fullscreen	specifica se l'applicazione girerà in finestra oppure a pieno schermo
stencilbuffer	abilita o meno l'uso dello stencil buffer per il disegno delle ombre
vsync	abilita o meno l'uso della sincronizzazione verticale a pieno schermo (il "vsync" appunto)
eventReceiver	l'oggetto che riceve gli "eventi" generati dal motore. Riprenderemo questo argomento in seguito.

Tabella 1

NOTA

AGGIUNGERE LA MESH ALLA SCENA

Una volta creata la mesh che desideriamo, è necessario aggiungerla alla scena da disegnare: questo viene fatto creando un "nodo" nella struttura ad albero della scena, che contenga la mesh. Si utilizza la funzione *addAnimatedMeshSceneNode()* che restituisce il puntatore al nodo, il quale viene salvato nella variabile "*node*".

IL CODICE ILLUSTRATO

```
IAnimatedMesh* mesh = smgr->getMesh(
                              "colonna.x");
IAnimatedMeshSceneNode* node =
  smgr->addAnimatedMeshSceneNode(mesh);
smgr->addCameraSceneNode(
                   0, vector3df(0,300,-400),
                       vector3df(0,100,0));
```

4 FINALMENTE SI DISEGNA

3 istruzioni per caricare il disegno della colonna dal file *colonna .x,* aggiungere l'oggetto mesh appena creato ad una scena e posizionare una videocamera che utilizzeremo in seguito.

```
while(device->run())          {
   driver->beginScene(true, true,
                    SColor(0,200,200,200));
   smgr->drawAll();
   driver->endScene();
}
```

5 FINITO!!!

4 istruzioni per creare un ciclo che mostri la scena che abbiamo disegnato fino a quando qualcuno non chiude la finestra o non preme *f4*.

```
device->drop();
return 0;
}
```

6 RIPULIAMO IL TUTTO

2 righe che rilasciano tutte le risorse allocate al programma, e restituiscono il computer lindo senza lasciare pezzi di memoria invasi da "spazzatura".

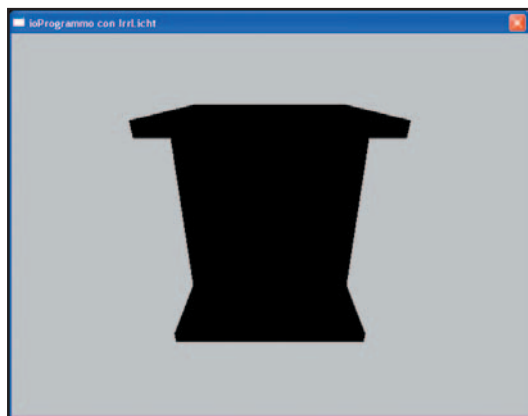



Fig. 1: *Un modello 3D visualizzato in una finestra*

relative all'interfaccia utente) devono essere effettuate all'interno del blocco delimitato dalle chiamate a funzione *beginScene()* e *endScene()*. In particolare la prima di queste si occupa di pulire lo schermo con un determinato colore passato come parametro. Il gestore della scena smgr è utilizzato per disegnare a schermo tutta la scena (per adesso solo la mesh della colonna), utilizzando la videocamera creata in precedenza, tramite la funzione *drawAll()*.

Nel **passo 6** del codice di esempio viene invocata la funzione drop() sul device, in modo da rilasciare tutte le risorse occupate dal programma. Come regola generale in IrrLicht ogni oggetto creato con una funzione di tipo *createX()* viene rilasciato con una *drop()*. Questo è particolarmente importante quando si ha a che fare con applicazioni in cui, tipicamente, l'ottimizzazione della gestione delle risorse è molto importante, come le applicazioni in 3D real-time.

ACCENDIAMO LE LUCI E POSIZIONIAMO LA VIDEOCAMERA

In verità, il risultato ottenuto fin qui non è tanto esaltante (almeno agli occhi di un profano!). Viene infatti disegnata a schermo la nostra colonna completamente nera, come in **Figura 1** difficilmente riconoscibile. Il problema in questo caso è l'assenza di luce. Creare una luce da aggiungere a una scena di *IrrLicht* è semplicissimo e si riduce nell'aggiungere una singola riga di codice. Inserendo:

```
smgr->addLightSceneNode(0, vector3df(0,300,-400),
SColor(0,200,200,200),400.0f,-1);
```

nel passo 4 del codice di esempio, si otterrà l'immagine di **Figura 2**. Come si vede la colonna è adesso illuminata ed è possibile distinguerne l'aspetto, in particolare le texture che la fanno sembrare costruita di mattoni. Tutto quello che abbiamo fatto è stato aggiungere un nodo di tipo "luce" alla scena, tramite la funzione *addLightSceneNode()*. I parametri accettati da questa funzione sono sono diachiarati in *Tabella 2.*

Fig. 2: *L'aggiunta di luce alla scena*

AGGIUNGIAMO UNA TEXTURE

Con la stessa facilità con cui abbiamo aggiunto una luce possiamo caricare una nuova texture per la colonna, in modo da farla apparire con un aspetto totalmente diverso. Aggiungendo ad esempio il se-

guente frammento in coda al **passo 4**:

```
if (node) {
        node->setMaterialTexture( 0,
driver->getTexture("mattoni_02.jpg"));
}
```

si cambia la texture della colonna sostituendo quella originale con quella del file "*mattoni_02.jpg*". Il risultato è visibile in **Figura 3**. Da notare come la variabile "node" salvata in precedenza sia utilizzata per tutte le operazioni riguardanti la mesh contenuta nel nodo. Attenzione ovviamente a non utilizzare "*node*" se questa è un puntatore non valido: è sempre opportuno eseguire un controllo tramite un blocco IF, come fatto, appunto, nell'esempio riportato.

***Fig. 3:* L'aggiunta di una texture all'oggetto "colonna"**

L'ultimo miglioramento che apporteremo al nostro programma lo renderà in qualche modo interattivo. Sostituiamo infatti alla videocamera fissa una videocamera controllata tramite mouse+tastiera. Non c'è da preoccuparsi però! Non dovremo impazzire con complessi calcoli riguardanti gli offset di spostamento del mouse o cose del genere: ci basterà sostituire alla riga:

```
smgr->addCameraSceneNode(0,
vector3df(0,300,-400), vector3df(0,100,0));
```

la seguente:

```
smgr->addCameraSceneNodeFPS();
```

Questo creerà un sistema di controllo simile a quello utilizzato nei videogiochi *First Person Shooter* (= FPS, da cui il nome della funzione) basato sui movimenti del mouse e delle frecce direzionali. Un esempio di come si possa adesso cambiare la visuale è visibile in **Figura 4**. È disponibile anche una funzione analoga chiamata *addCameraSceneNodeMaya()* che aggiunge un sistema di controllo simile a quello utilizzato nel software di modellazione tridimensionale Maya. Ovviamente è possibile sviluppare un sistema di controllo della videocamera (o in generale della scena) raffinato quanto si vuole, partendo dalle primitive offerte da IrrLicht. La presenza di questi controlli predefiniti, tuttavia, è di enorme aiuto nelle fasi di test del software o in quelle di prima stesura, in quanto consente di avere pieno controllo su quello che succede nella scena, evitando di sprecare risorse nella gestione del sistema di controllo.

CONCLUSIONI

Abbiamo visto in questo primo articolo i fondamenti di funzionamento del motore grafico IrrLicht. Il caricamento delle mesh da file, la creazione di telecamere e luci dinamiche e il ciclo fondamentale di un programma IrrLicht (con le relative *beginScene()* e *endScene()*) sono tra le cose basilari, ma anche più importanti, che è bene conoscere per padroneggiare al meglio queste librerie. Come abbiamo già avuto modo di pregustare, inoltre, l'aggiunta di caratteristiche avanzate è davvero immediata (si pensi solo alla differenza tra il codice dell'esempio iniziale e l'applicazione "modificata" finale), quindi l'unico limite è rappresentato dalla creatività e dalla competenza del programmatore.

Nei prossimi articoli approfondiremo aspetti quali la gestione della scena 3D, l'utilizzo di complesse mappe organizzate in file di tipo .bsp, nonché il wrapping del motore con la piattaforma .NET. Nel frattempo potete dare un'occhiata più approfondita al codice completo di questo articolo, che è stato inserito nel CD insieme alle texture utilizzate e al modello 3D della colonna.

Alfredo Marroccelli

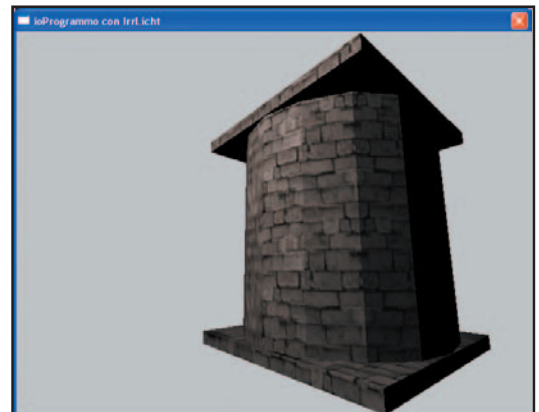
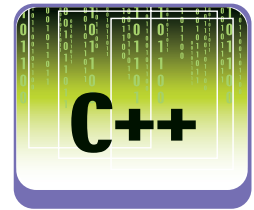



***Fig. 4:* Possiamo cambiare la visuale utilizzando mouse e tastiera**

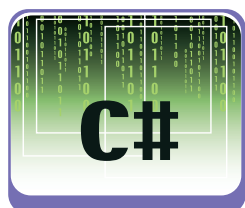
VISUALE
È disponibile una funzione chiamata *addCameraSceneNode Maya()* che aggiunge un sistema di controllo simile a quello utilizzato nel software di modellazione tridimensionale Maya.

parent	il "nodo genitore" di questo nodo. Approfondiremo in seguito la struttura di una scena di IrrLicht, per adesso utilizziamo un valore nullo per i nodi genitori.
position	la posizione della luce dinamica creata
color	il colore irraggiato dalla luce
radius	il "raggio di influenza" della luce (provando a mettere un valore inferiore la scena risulterà più buia, al contrario sarà più luminosa con un valore più alto)
id	l'identificatore del nodo

Tabella 2: I parametri accettati da addLightSceneNode()

Scriviamo il codice C# di un semplice VideoGioco interattivo

Astronavi contro asteroidi

In questo articolo caricheremo un modello 3D in memoria, controlleremo le capacità della scheda video, accenderemo le luci e impareremo a gestire le collisioni

Utilizza questo spazio per le tue annotazioni

Conoscenze richieste
C#; .NET FrameWork; Programmazione ad oggetti

Software
Microsoft Visual Studio .NET 2003, DirectX 9 SDK installati su una piattaforma Microsoft (Windows 2000, Windows XP, Windows 2003 Server)

Impegno

Tempo di realizzazione

Il modo più semplice per creare ambienti 3D è, caricare i modelli 3D da un file esterno (in genere con estensione .X) creato con programmi di modelling come LightWave o 3D Studio Max ed inviare direttamente i dati su schermo. Questi modelli vengono quindi prima caricati in oggetti DirectX chiamati *Mesh* e poi renderizzati su schermo. Creare una istanza di una Mesh è semplicissimo:

```
private Mesh LaMiaMesh = null;
```

Per caricare una Mesh da un file basta poi richiamare la funzione statica *FromFile* dell'oggetto Mesh come segue:

```
Mesh LaMiaMesh = Mesh.FromFile(NomeFile,
              MeshFlags.Managed, device, out materiali);
```

Il primo parametro non è altro che il nome del file .X da caricare in memoria, il secondo parametro indica come e dove la risorsa viene caricata. La **Tabella 1** riporta l'elenco completo di tutti i valori dell'oggetto *MeshFlags*. Il terzo parametro della funzione è l'ormai consolidato oggetto *Device* su cui verrà renderizzata la Mesh. L'ultimo parametro è una variabile, in uscita, di tipo *ExtendedMaterial[]*. Questo tipo di variabile è come il normale oggetto *Material* già affrontato nel precedente articolo, che contiene però anche la stringa con il nome della texture da caricare a da applicare alla Mesh.

Fig. 1: **Un VideoGioco commerciale**

CARICHIAMO UN MODELLO 3D IN MEMORIA

Per caricare una Mesh potremmo quindi scrivere una funzione come questa:

```
public static Mesh CaricaMesh(Device device, string
        file, ref Material[] meshMaterials, ref Texture[]
                                    meshTextures){
 ExtendedMaterial[] mtrl;
 // Dichiariamo un array di ExtendedMaterial, in
 // quanto una Mesh può essere composta da
 // più materiali e texture.
 Mesh tempMesh = Mesh.FromFile(file,
              MeshFlags.Managed, device, out mtrl);
 // Carichiamo la Mesh in memoria
 if ((mtrl != null) && (mtrl.Length > 0)) {
 // Se c'è almeno un materiale procediamo
 // nell'elaborazione della Mesh
   meshMaterials = new Material[mtrl.Length];
 // Dichiaro il numero di materiali presenti nella Mesh
   meshTextures = new Texture[mtrl.Length];
   // Dichiaro il numero di texture presenti nella Mesh
   for (int i = 0; i < mtrl.Length; i++) {
   // Ciclo per ogni materiale della Mesh
     meshMaterials[i] = mtrl[i].Material3D;
   // Salvo in meshMaterials il materiale corrente
     if ((mtrl[i].TextureFilename != null) &&
          (mtrl[i].TextureFilename != string.Empty)) {
       meshTextures[i] = TextureLoader.FromFile(
          device, @"..\..\" + mtrl[i].TextureFilename);
   // Se a questo materiale è associata una Texture la
   // carico in memoria e la salvo dentro meshTextures
     }}}
```

```
  return tempMesh; //Ho finito di caricare la Mesh
}
```

Come prima istruzione dichiariamo l'array *mtrl* di tipo *ExtendedMaterial* poiché una mesh può essere formata da uno o più materiali. A questo punto siamo pronti per caricare la mesh in memoria dentro l'oggetto *tempMesh* attraverso la funzione *FromFile* appena vista, e, se la mesh è valida (il che equivale a controllare che ci sia almeno un materiale), si procede con la dichiarazione dei materiali e delle texture necessari. A questo punto ciclo per ogni coppia materiale/texture, ossia per ogni Subset della mesh, dove oltre a salvare in *meshMaterials* il materiale corrente carico anche la texture quando presente (se non è presente una texture lascio il valore di default *null*). Finito il ciclo ho una mesh valida in memoria che passo come valore di ritorno della funzione.

DISEGNIAMO SULLO SCHERMO

Una volta caricato un oggetto Mesh in memoria, dovremmo scrivere una funzione per disegnarlo sul device:

```
private void DisegnaMesh(Device device, Mesh MyMesh)
{  for (int i = 0; i < meshMaterials.Length; i++){
        device.Material = meshMaterials[i];
        device.SetTexture(0, meshTextures[i]);
        MyMesh.DrawSubset(i); }
}
```

Cicliamo ogni materiale della Mesh (ognuno di essi viene impostato come materiale corrente del device), lo stesso per la texture associata al materiale in questione, ed infine viene disegnata la parte della Mesh che usa quel materiale e quella texture, attraverso la funzione *DrawSubset* dell'istanza della Mesh. È importante precisare che questo codice è assolutamente generico, il che significa che con queste due funzioni possiamo disegnare qualsiasi oggetto contenuto in un file *.X*, da un'astronave (come faremo fra poco) ad un essere umano, e così via.

SFRUTTIAMO AL MASSIMO LA SCHEDA VIDEO

La procedura di inizializzazione di un videogioco dovrebbe essere un po' più accurata di quelle viste fino ad ora, in quanto si dovrebbe cercare di sfruttare al meglio le caratteristiche hardware offerte dalla scheda grafica su cui gira il gioco. Vediamo insieme come fare.

```
public void InitializeGraphics() {
      PresentParameters presentParams = new
                                   PresentParameters();
      presentParams.Windowed = true;
      presentParams.SwapEffect = SwapEffect.Discard;
      presentParams.AutoDepthStencilFormat =
                                      DepthFormat.D16;
      presentParams.EnableAutoDepthStencil = true;
     int adapterOrdinal=Manager.Adapters.Default.Adapter;
     CreateFlags flags =
                    CreateFlags.SoftwareVertexProcessing;
     Caps caps = Manager.GetDeviceCaps(
                    adapterOrdinal, DeviceType.Hardware);
     if (caps.DeviceCaps.
          SupportsHardwareTransformAndLight) flags =
                    CreateFlags.HardwareVertexProcessing;
     if (caps.DeviceCaps.SupportsPureDevice) flags |=
                                  CreateFlags.PureDevice;
     device = new Device(adapterOrdinal,
          DeviceType.Hardware, this, flags, presentParams);
     device.DeviceReset += new
                     EventHandler(this.OnDeviceReset);
     this.OnDeviceReset(device, null);
}
```

Prima di tutto creiamo e impostiamo la struttura di presentation parameters come fatto fino ad ora. Poi dichiariamo una variabile flags di tipo *CreateFlags* che impostiamo con il valore *SoftwareVertexProcessing* supportato da ogni scheda grafica in commercio. Quindi si testa se l'hardware supporta l'accelerazione per le trasformazioni e le luci, ed in caso positivo andiamo a sovrascrivere il valore di flags con *HardwareVertexProcessing*. Infine se l'hardware supporta anche un PureDevice aggiungiamo questo valore a flags: a questo punto siamo pronti per creare un device che sfrutterà al meglio le capacità di ac-

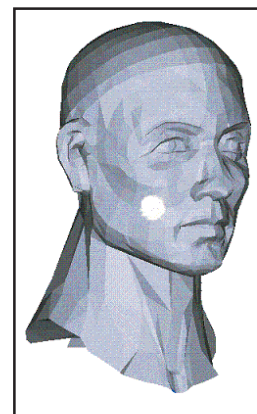
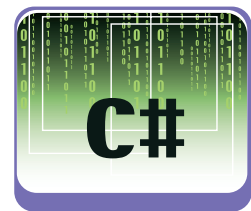



Fig. 2: Un esempio di mesh

Parametri	Valore
MeshFlags.DoNotClip	Non esegue il Clip nel vertex e index buffer
MeshFlags.Dynamic	E' l'insieme di IbDynamic e VbDynamic
MeshFlags.IbDynamic	Usa il flag Usage.Dynamic per l'index buffer
MeshFlags.IbManaged	Usa la memoria Pool.Managed per gli index buffer
MeshFlags.IbSoftwareProcessing	Usa il flag Usage.SoftwareProcessing negli index buffer
MeshFlags.IbSystemMem	Usa la memoria Pool.SystemMemory per gli index buffer
MeshFlags.IbWriteOnly	Usa il flag Usage.WriteOnly per gli index buffer
MeshFlags.VbDynamic	Usa il flag Usage.Dynamic per il vertex buffer
MeshFlags.VbManaged	Usa la memoria Pool.Managed per il vertex buffer
MeshFlags.VbSoftwareProcessing	Usa il flag Usage.SoftwareProcessing nel vertex buffer
MeshFlags.VbSystemMem	Usa la memoria Pool.SystemMemory per il vertex buffer
MeshFlags.VbWriteOnly	Usa il flag Usage.WriteOnly per il vertex buffer
MeshFlags.Managed	E' l'insieme di IbManaged e VbManaged
MeshFlags.Npatches	Usa il flag Usage.NPatches nel vertex ed index buffer
MeshFlags.Points	Usa il flag Usage.Points nel vertex ed index buffer
MeshFlags.RtPatches	Usa il flag Usage.RtPatches nel vertex ed index buffer
MeshFlags.SoftwareProcessing	È l'insieme di IbSoftwareProcessing e VbSoftwareProcessing
MeghFlags.SystemMemory	È l'insieme di IbSystemMem e VbSystemMem
MeshFlags.Use32Bit	Usa indici a 32-bit nell'index buffer (sconsigliato)
MeshFlags.UseHardwareOnly	Viene elaborata solo con l'accelerazione Hardware

Tabella 1: Tutti i valori dell'oggetto MeshFlags

celerazione hardware supportate dalla scheda grafica.

GESTIAMO LE LUCI

È importante anche aggiornare e rendere più stabile la nostra funzione *OnResetDevice,* aggiungendo un controllo per testare se l'hardware con cui abbiamo creato il Device supporta il numero di luci attive che intendiamo usare. Per far ciò dobbiamo controllare le proprietà dell'oggetto *Device.DeviceCaps*. Nel caso specifico volendo usare due luci direzionali nel nostro videogioco controlliamo se il valore di MaxActiveLights sia almeno maggiore di uno e se il valore di *VertexProcessingCaps.SupportsDirectionalLights* sia true: in caso positivo procediamo con l'implementazione standard delle luci. Se la scheda grafica non dovesse supportare due luci direzionali, attiviamo allora una semplice luce ambientale per illuminare la scena (la luce ambientale è disponibile su qualsiasi hardware). È chiaro che così rinunciamo però agli effetti più realistici che le due luci incrociate avrebbero prodotto sul gioco.
Il codice da inserire in *OnResetDevice* sarà quindi:

```
if ((device.DeviceCaps.VertexProcessingCaps.
                    SupportsDirectionalLights) &&
  (device.DeviceCaps.MaxActiveLights > 1)) {
    device.Lights[0].Type = LightType.Directional;
    device.Lights[0].Diffuse = Color.White;
    device.Lights[0].Direction = new Vector3(1, -1, -1);
    device.Lights[0].Commit();
    device.Lights[0].Enabled = true;
    device.Lights[1].Type = LightType.Directional;
    device.Lights[1].Diffuse = Color.White;
    device.Lights[1].Direction = new Vector3(-1, 1, -1);
    device.Lights[1].Commit();
    device.Lights[1].Enabled = true; }
  else
  {device.RenderState.Ambient = Color.White;}
```

MUOVIAMO LE ASTRONAVI

Avremo bisogno di una mesh che rappresenta l'astronave che pilotiamo, una mesh per lo sfondo ed una per gli asteroidi che incontriamo nello spazio. Lo scopo del gioco sarà abbastanza semplice: evitare le collisioni con gli asteroidi mentre pilotiamo l'astronave. Come possiamo fare ciò? È davvero il caso di dire che è più semplice a farsi che a dirsi, in quanto ci basterà creare una classe per la mesh dell'astronave ed una per la mesh dell'asteroide. Nella prima implementeremo la gestione dei comandi: se l'input da tastiera è di andare a destra, disegneremo l'astronave più a destra, e analogamente se l'input è

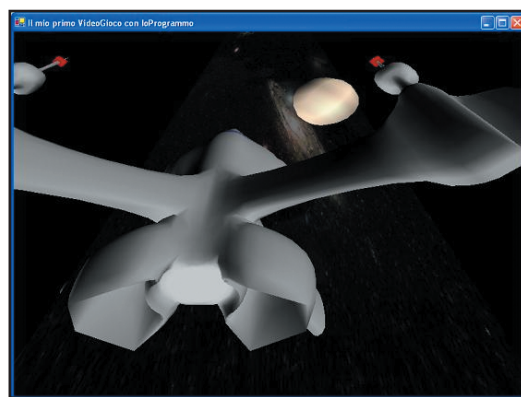



Fig. 3: **Il nostro gioco finito**

di andare a sinistra. Il tutto viene realizzato semplicemente incrementando o decrementando la variabile *PosizioneAstronave* che rappresenta appunto la posizione dell'astronave lungo l'asse X (ossia quando è a destra e quando a sinistra sul nostro schermo). Una volta aggiornata questa variabile, sarà sufficiente usare una matrice di traslazione per spostare l'astronave a destra o a sinistra. La seguente funzione della classe *Astronave* mostra quando detto fino ad ora.

```
public void Draw(Device device) {
    device.Transform.World = Matrix.Translation(
            PosizioneAstronave, Altezza, Profondita);
    for (int i = 0; i < AstroMaterials.Length; i++){
        device.Material = AstroMaterials[i];
        device.SetTexture(0, AstroTextures[i]);
        AstroMesh.DrawSubset(i); } }
```

MUOVIAMO GLI ASTEROIDI

Per quanto riguarda la classe che contiene la mesh dell'asteroide, non avendo a disposizione un file .X per caricare il modello, useremo una delle funzioni statiche dell'oggetto Mesh come segue:

```
asteroideMesh = Mesh.Sphere(device, Raggio,
      Utility.Rnd.Next(10)+2, Utility.Rnd.Next(10)+2);
```

Il primo parametro della funzione è il device usato nell'applicazione, il secondo è un float che rappresenta il raggio della sfera che verrà creata. Come secondo e terzo parametro (che indicano quante facce e quanti vertici avrà la sfera) passiamo un numero relativamente basso e casuale, in modo di avere asteroidi abbastanza spigolosi (poco "sferici") e di non averli tutti uguali fra loro. Questa classe conterrà anche la funzione booleana che controlla se è avvenuta una collisione

```
public bool Collisione(float fLocation, float fDiameter)
```

NOTA

IL NUMERO DI LUCI SUPPORTATO DA DIRECTX

DirectX supporta fino ad 8 luci attive contemporaneamente, ma non è detto che questo numero sia altresì supportato anche dalla scheda grafica

LE FUNZIONI STATICHE DELLA CLASSE MESH

La classe Mesh presenta delle funzioni statiche che ritornano dei semplici oggetti pronti ad essere renderizzati su schermo, senza bisogno quindi di doverli caricare da file .X. Queste sono ***Mesh.Box*** che ritorna la mesh di un cubo, ***Mesh.Cylinder, Mesh.Polygon, Mesh.Torus, Mesh.Teapot*** e ***Mesh.Sphere*** che ritornano rispettivamente la mesh di un cilindro, di un poligono, di un toro (ciambella), di una teiera (oggetto simbolico delle rappresentazioni con DirectX) e una sfera.

A questa verrà passata la posizione e il diametro dell'astronave. Per quanto riguarda la posizione dell'astronave non abbiamo problemi, ma come facciamo ad ottenere il diametro dato che la mesh è stata caricata da un file esterno? Avremo bisogno di richiamare una funzione *ComputeBoundingSphere* della libreria *D3DX* e di inserirla nel costruttore della classe come segue:

```
public Astronave(Device device){
  AstroMesh = MyGame.LoadMesh(device, @"..\..\
         Star.x", ref AstroMaterials, ref AstroTextures);
  VertexBuffer vb = AstroMesh.VertexBuffer;
  try {
    GraphicsStream stm = vb.Lock(0, 0, LockFlags.None);
    Vector3 center;
    float radius = Geometry.ComputeBoundingSphere(
                        §stm, AstroMesh.NumberVertices,
                    AstroMesh.VertexFormat, out center);
    DiametroAstronave = (radius * 2); }
  finally {
    vb.Unlock();
    vb.Dispose(); } }
```

Dopo aver caricato la nostra mesh attraverso la nostra *LoadMesh* dichiariamo un *VertexBuffer*, *vb*, al quale associamo il *VertexBuffer* della mesh. A questo punto, all'interno di un blocco try, eseguiamo un *Lock* del VertexBuffer per ottenere un oggetto *GraphicsStream*, stm, come è richiesto dalla funzione *ComputeBoundingSphere*. Lo stesso vale per il vettore center che non useremo ma è comunque anch'esso richiesto dalla funzione. Possiamo ora finalmente richiamare questa funzione che restituisce il raggio della sfera che circonda la mesh, espresso come sempre da un valore float. Ottenuto in questo modo il raggio, basta moltiplicarlo per due per ottenere il diametro della sfera che avvolge l'astronave. Per concludere, dentro un blocco finally, eseguiamo l'*Unlock* e poi il *Dispose* dell'oggetto *VertexBuffer* che abbiamo usato.
A questo punto non resta che analizzare due funzioni importanti. La prima è quella che disegna il nostro sfondo spaziale.

```
private void DisegnaSfondo(float x, float y, float z) {
  device.Transform.World = Matrix.Translation(x, y, z);
  for (int i = 0; i < SfondoMaterials.Length; i++) {
    device.Material = SfondoMaterials[i];
    device.SetTexture(0, SfondoTextures[i]);
    SfondoMesh.DrawSubset(i); }
}
```

A questa funzione passiamo semplicemente le coordinate *x*, *y* e *z* dello sfondo che useremo per disegnarlo esattamente dove vogliamo. Poi non facciamo altro che richiamare le ormai note righe di codice che disegnano una qualsiasi mesh.

RITOCCHI FINALI

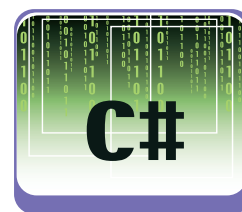
L'ultima funzione che vale la pena approfondire è l'override della funzione *OnPaint* del form sul quale sviluppiamo il nostro videogioco.

```
protected override void
OnPaint(System.Windows.Forms.PaintEventArgs e) {
   AggiornaTutto();
   device.Clear(ClearFlags.Target | ClearFlags.ZBuffer,
                                    Color.Black, 1.0f, 0);
   device.BeginScene();
   DisegnaSfondo(0.0f, 0.0f, Prof0);
   DisegnaSfondo(0.0f, 0.0f, Prof1);
    astro.Draw(device);
    foreach(Asteroide o in asteroidi) {
      o.Draw(device);
    }
    device.EndScene();
    device.Present();
    this.Invalidate();
}
```

Come prima cosa richiamiamo la funziona *AggiornaTutto()* nella quale gestiamo l'aumento costante di velocità nel gioco. Fatto ciò puliamo il Buffer del device impostando il colore nero come sfondo ed iniziamo quindi il rendering con *BeginScene()*. Disegneremo lo sfondo due volte a delle profondità diverse aggiornate in modo ciclico, così da creare un effetto di astronave che si muove in avanti, quando in realtà è lo sfondo che viene disegnato facendolo scorrere costantemente (anche la gestione delle due variabili *Prof0* e *Prof1*, con le quali otteniamo questo scorrimento, è contenuta in *AggiornaTutto*).
A questo punto richiamiamo il metodo *Draw* della classe che contiene l'astronave in modo da renderizzare la relativa mesh, e facciamo lo stesso per ogni asteroide contenuto nell'array asteroidi. Fatto ciò abbiamo rappresentato tutti i nostri oggetti del gioco e possiamo richiamare la *EndScene()* e la *Present()*.
Concludiamo con la funzione *this.Invalidate()* in modo da richiamare nuovamente l'evento *OnPaint()* del form, e tenere così sempre vivo il ciclo di aggiornamento e disegno del nostro gioco.

IL FUNZIONAMENTO DEL PROGRAMMA

Con il tasto "4" (o freccia a sinistra) si sposta verso sinistra l'astronave, con il tasto "6" (o freccia a destra) si sposta verso destra l'astronave. Con il tasto 5 si ferma l'astronave nella posizione attuale.
Con ESC si esce dal gioco: buon divertimento!
Per qualsiasi informazione scrivetemi pure a *carlozoffoli@ioprogrammo.it*

Carlo Federico Zoffoli

CONVERTIRE I FILE IN .X

L'SDK di DirectX include molte utility per convertire i file prodotti da programmi di modelling 3D in file .X. Si possono quindi tranquillamente usare queste applicazioni per importare nel proprio software file prodotti anche da programmi che non supportano il salvataggio dei disegni in file con estensione .X.

FILL RATE

È la velocità alla quale una scheda grafica riesce a renderizzare i pixel, ed è generalmente misurata in milioni di pixel al secondo (*Megapixels/sec*). Le GPU con un elevato fill rate possono quindi rappresentare con una risoluzione migliore, con più colori e ad un più elevato frame rate le scene disegnate dalle vecchie GPU meno potenti.

Qualche accorgimento per rendere la nostra MIDlet più "godibile"

SMS "gratis" dal cellulare

parte seconda

Miglioriamo l'applicazione che ci permette di inviare messaggi gratis. Aggiungiamo qualcosa di utile (un contacaratteri e l'elenco dei destinatari recenti) e qualcosa di accattivante graficamente

In questo articolo, il secondo riguardante l'invio di SMS via gprs da telefonino, cercheremo di raffinare il più possibile l'interfaccia utente, studiando soluzioni che migliorino la "user experience" sia dal punto di vista funzionale, sia dal punto di vista estetico. Cercheremo, insomma, di rendere il nostro programmino simile ad un'applicazione "commerciale", staccandoci dal sapore "casereccio" che, inevitabilmente, le nostre creature assumono, specialmente quelle strutturalmente semplici, come SmsSender. La nostra applicazione deve essere in grado di "girare" su un cellulare dal display monocromatico, con poca risoluzione, così come su un evoluto smartphone di terza generazione, magari dotato della banda offerta da UMTS, con un grande e nitido display in grado di visualizzare migliaia e migliaia di colori. Ma con qualche accorgimento riusciremo a rendere più "godibile" la nostra amata applicazione, che già solo nella sua utilità troverebbe motivo di esistere, approfittandone per studiare qualche costrutto più avanzato, che potrà tornarci utile quando vorremo sviluppare qualcosa di più sostanzioso.

Fig. 1: ***Rivediamo l'infrastruttura del nostro progetto. Rimarrà invariata, le modifiche avverranno sull'interfaccia utente della midlet***

CONTIAMO I MESSAGGI

Se avete utilizzato l'applicazione in questo mese, avrete sicuramente sentito la mancanza di un meccanismo che ci segnali quanti messaggi abbiamo inviato nella giornata corrente. Vodafone Italia ci offre, infatti, un numero limitato (10 al giorno) di SMS da inviare attraverso il servizio *"sms via mail"*, gratuito per tutti gli iscritti al suo portale (*190.it*).

Sarebbe parecchio utile, quindi, sapere in anticipo se abbiamo superato la soglia quotidiana, per evitare figuracce con i destinatari "mancati" dei nostri messaggini. Ad ogni invio dovremo, quindi, incrementare il contatore che andrà salvato su memoria persistente e resettato all'inizio di ogni giorno. Ma come possiamo fare ad implementare questo meccanismo? Un approccio valido potrebbe essere quello di creare un oggetto *ultimoInvio* che includa il numero progressivo dell'ultimo invio e il *timestamp* da cui ricavare il giorno in cui è stato inviato. Così, ad ogni invio, basta caricare questi dati e se il giorno è cambiato, azzerare il contatore, altrimenti semplicemente incrementarlo. In J2me esiste un unico modo per accedere alle varie parti della data, cosa che non piacerà a chi ha già manipolato le date in java, sicuramente con il metodo *"officially deprecated" get()* di *java.util.Date*. Il metodo attualmente "consigliato" dalle API java, l'unico presente nell'implementazione microedition, è, infatti, parecchio più macchinoso: bisogna istanziare un oggetto *Calendar*, fornirgli attraverso il metodo *setTime(Date date)* l'attuale *timestamp* e ottenere, finalmente, i vari *FIELD* della data invocando sul *Calendar* il metodo *get(int field)* più volte (una per l'anno, una per il mese, una per il giorno). Per non dover inventare strani cicli di controllo preferiamo avere un solo intero, facilmente confrontabile con un altro, che contenga la data odierna. Il formato più adatto è sicuramente: *AAAAMMGG* tipo *20041014*, che il confronto aritmetico riconosce, giustamente, minore di *20041018*. Vediamo come ottenere tale intero:

```
public int getDataFormattata(Date data)
{
  Calendar calendario= Calendar.getInstance();
  calendario.setTime(data);
```

```
  int dataformattata=0;
  //Aggiungo il giorno del mese
  dataformattata += calendario.get(
                        Calendar.DAY_OF_MONTH);
  //Aggiungo il mese
  dataformattata += 100*(calendario.get(
                        Calendar.MONTH));
  //Aggiungo l'anno
  dataformattata += 10000*(calendario.get(
                        Calendar.YEAR));
  return dataformattata;
}
```

Occorre sottolineare che dato che *Calendar* è una classe astratta, non possiamo costruirla ma solo ottenerne un'istanza attraverso il metodo statico *Calendar.getInstance* e che il mese non è indicato, come ci aspetteremmo, a partire da 01 per gennaio, ma a partire da 00, come per gli indici di un array. Quindi il 5 novembre 2004, sarà nel formato che abbiamo ottenuto: *20041005* e non *200411205* come potremmo pensare. Ad ogni modo, attraverso:

```
getDataFormattata(new Date());
```

otteniamo un intero che rappresenta la data al momento della chiamata, che possiamo confrontare con la data salvata precedentemente.
Analizziamo la classe *ultimoInvio* che utilizziamo per gestire il contatore:

```
public class ultimoInvio{
  int dataui;
  int quantiui;
  public ultimoInvio(int quantiui, int dataui){
    this.dataui=dataui;
    this.quantiui=quantiui;
  }
  public String toString(){
    return ""+dataui+";"+quantiui;
  }
}
```

Il metodo *toString()* restituisce una stringa opportunamente formattata (data e contatore divisi da un ";") utile per il salvataggio, dato che il metodo *addRecord* di *RecordStore* vuole un array di byte come argomento, array che otteniamo facilmente da una *String:*

```
public void salvaUltimoInvio(ultimoInvio aUltimoInvio){
  RecordStore recStoreui = null;
  String REC_STOREui = "SmsSenderTime";
  byte[] rec = aUltimoInvio.toString().getBytes();
  try{
    recStoreui = RecordStore.openRecordStore(
                        REC_STOREui, true );
    recStoreui.addRecord(rec, 0, rec.length);
    recStoreui.closeRecordStore();
  }catch (Exception e){} }
public  ultimoInvio caricaUltimoInvio(){
  RecordStore recStoreui = null;
  String REC_STOREui = "SmsSenderTime";
  String recui="null";
  try{
    recStoreui = RecordStore.openRecordStore(
                        REC_STOREui, true );
    byte[] recData = new byte[5];
    int len;
    int r= recStoreui.getNumRecords();
    for (int i = 1; i <= r; i++){
      if (recStoreui.getRecordSize(i) > recData.length)
        recData = new byte[recStoreui.getRecordSize(i)];
        len = recStoreui.getRecord(i, recData, 0);
        recui=new String(recData, 0, len); }
      recStoreui.closeRecordStore();
    }catch (Exception e){
      return new ultimoInvio(0,0); }
    String data="";
    String quanti="";
    int pv= recui.indexOf(";");
    if (pv>0) {
      data= recui.substring(0,pv);
      quanti=recui.substring(pv+1); }
     else {
       quanti="0";
       data="0"; }
     return new ultimoInvio( Integer.parseInt(
                        data),Integer.parseInt(quanti));
}
```

Questi due metodi, utilizzati in avvio e chiusura del programma, tengono il contatore sempre aggiornato. Caricando nello *startApp* la data dell'ultimo invio e confrontandola con l'attuale, potremo utilizzare il contatore salvato o, se la data è cambiata, azzerarlo:

```
ultimoInvio last= caricaUltimoInvio();
if (last.dataui==getDataFormattata(new Date()))
  contaMessaggi=last.quantiui;
else
  contaMessaggi=0;
```

In *destroyApp*, invece, che viene invocato alla chiusura della midlet, salveremo la data attuale e il numero di messaggi, dopo aver effettuato un ultimo controllo per evitare bug in caso di utilizzo a cavallo della mezzanotte!

```
public void destroyApp( boolean unconditional ) {
  ultimoInvio uia;
  ultimoInvio last= caricaUltimoInvio();
  if (last.dataui==getDataFormattata(new Date())) {
    uia= new ultimoInvio(getDataFormattata(
                        new Date()),contaMessaggi); }
   else
```

In una midlet, i metodi *startApp* e *destroyApp* sono invocati all'avvio e alla chiusura dell'applicazione. Possiamo effettuare le varie operazioni di salvataggio (del *timestamp* per esempio) in quest'ultimo metodo.

L'accesso a dati "privati" come la rubrica memoria sms, la rubrica o il filesystem del cellulare non è permesso ad una applicazione j2me. Questo perchè la midlet è eseguita in una "sandbox" che la isola dal resto del cellulare per motivi di sicurezza. Implementazioni dei produttori possono sopperire a queste mancanze, per esempio: *com.siemens.mp.io.File* presente su alcuni terminali Siemens

```
    uia= new ultimoInvio(getDataFormattata(
                                    new Date()),0);
    salvaUltimoInvio(uia); }
```

UNA MINI-RUBRICA

La mancanza più grave della nostra midlet è sicuramente l'impossibilità di accedere alla rubrica del cellulare (per motivi di sicurezza, imposti dalle api *midp*) e dover ogni volta inserire il numero. Solitamente i nostri messaggi sono inviati sempre alle stesse persone quindi una lista di massimo 10 numeri sarà più che sufficiente. Vediamo come implementarla: utilizzeremo un metodo *addNumeriRecenti*, richiamato prima dell'invio del messaggio, che inserisce nella lista, se non è già presente, il destinatario. La lista è salvata in memoria e ricaricata ad ogni uso, con i soliti metodi per la scrittura e lettura su RMS. Vogliamo soffermarci invece sul modo in cui possiamo visualizzare e scegliere dalla lista il destinatario. Le api *midp* 1.0 mettono a disposizione una classe derivata da *Screen*, proprio per permettere la scelta da una lista, di uno o più valori. Questa classe, *List*, permette il controllo dell'elemento scelto attraverso il metodo *getSelectedIndex*, da invocare nel *CommandListener* associato. Ma vediamo direttamente nel codice come utilizzare la *List:*

```
listarecenti= new List("Numeri recenti",List.IMPLICIT);
//possiamo aggiungere, già nel codice, un numero che
//sappiamo di nostro frequente utilizzo:
//listarecenti.append("3401234567",null);
//Carichiamo la lista
String[] sArray=getNumeriRecenti();
for(int i = 0; i < sArray.length; i++){
   //appendiamo tutti gli elementi
   listarecenti.append(sArray[i],null); }
// "d" è l'istanza del Display
d.setCurrent(listarecenti);
selectCommand = new Command(
                        "Scegli", Command.ITEM, 1);
listarecenti.addCommand(selectCommand);
listarecenti.setCommandListener(this);
```

Nel *CommandAction* gestiremo l'utilizzo del numero recente:

```
if (c = = selectCommand){
   nrecente=listarecenti.getString(
                  listarecenti.getSelectedIndex()); }
```

Ultimo accorgimento: per limitare il numero di destinatari contenuti nella lista, possiamo, nel metodo di lettura *getNumeriRecenti* ciclare fino ad un massimo di 10 volte:

```
int totale=java.lang.Math.min(
                        recStorenr.getNumRecords(),10);
 for (int i = 1; i <= totale;i++){
  ....
}
```

Ovviamente il metodo per il salvataggio, *addNumeriRecenti* salverà i mittenti restituiti da quel metodo (massimo 10,quindi) , in modo da limitare anche la quantità di memoria utilizzata, che in un cellulare non deve mai essere eccessiva.

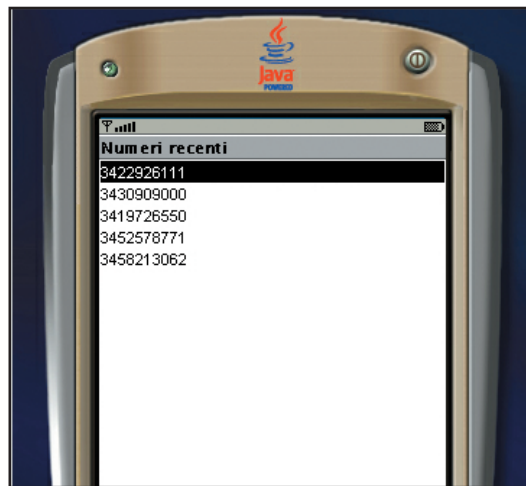



Fig. 2: La lista di numeri più utilizzati. Il numero scelto viene restituito alla form di inserimento

NOTA

Un'alternativa a Java, per lo sviluppo di applicazioni mobili, è implementare direttamente nel linguaggio nativo del terminale, cosa possibile sui vari smartphone. La cosa non è facilissima, dato che il linguaggio da usare (Symbian C, per gli smartphone Series 60 e UIQ) è di levello ben più basso rispetto a Java. Il beneficio è il controllo dell'intero cellulare.

MESSAGGI LUNGHI

Gli SMS, sono per definizione, messaggi brevi e solitamente la loro lunghezza è limitata a 160 caratteri. Quelli inviati con il servizio offertoci dalla Vodafone, sono ancora più brevi e dipendenti dalla lunghezza dell'indirizzo mail utilizzato come mittente.
Solitamente abbiamo a disposizione massimo 120 caratteri, che ci portano di frequente, ad inviare più di un messaggio. Vogliamo automatizzare quindi il processo di "spezzettamento" dei messaggi, in modo da non dover controllare il contatore di caratteri in continuazione. Per semplicità divideremo, direttamente prima dell'invio del messaggio, il testo in tante parti da 120 caratteri ed effettueremo più *http GET* alla nostra servlet. Per sapere di quanti caratteri è la stringa da inviare, utilizzeremo il metodo *length()* di *String* e taglieremo con *substring* 120 lettere alla volta, fino ad esaurire il buffer. Volendo potremmo fare parti più piccole per terminare ogni singolo messaggio con qualcosa tipo *[CONTINUA]* o dei punti di sospensione, ma i messaggi arriveranno in sequenza ravvicinata, e quei caratteri andrebbero sprecati, non essendo necessari per capire che si tratta un messaggio lungo.
Ecco come possiamo trasformare il metodo *send():*

```
public void send() {
```

```
    inviando = new Form("Invio in corso");
    inviando.addCommand(exit);
    inviando.setCommandListener(this);
    inviando.append("Connessione in corso..");
    d.setCurrent(inviando);
    String mess=mex.getString();
    String n=num.getString();
    addNumeriRecenti(n);
    torna = new Command("Ritorna", Command.OK, 1);
    String abuffer= mess;
    int i=1;
    while (abuffer.length()>=120){
      i++;
      mess= abuffer.substring(0,120);
      abuffer=abuffer.substring(121);
      URL=URL + "?text=" + mess +"&to="
                              + n+"&from="+MailMittente;
      URL=urlEncode(URL);
      String buffer = "\n";
      HttpConnection c = null;
      try {
        c = (HttpConnection)Connector.open(URL);
        InputStream is = c.openInputStream();
        int ch;
        while ((ch = is.read()) != -1) {
            buffer=buffer+( (char)ch ); }
        inviando.append("Messaggio "+i+": "+buffer);
        is.close();
        c.close(); }
      catch ( IOException e ) {
        inviando.append(e.toString()); } }
    inviando.addCommand(torna);
}
```

Questa soluzione è sicuramente la più veloce da implementare, ma contraddice uno degli obbiettivi fondamentali della nostra applicazione: l'economia di dati scambiati. Fare più richieste brevi invece di una sola, più lunga, non è conveniente sotto questo punto di vista. Ma una sola richiesta di tipo *Get* non è possibile, per la limitazione sulla lunghezza massima della Url, che è di 256 caratteri. Avremmo dovuto effettuare una richiesta di tipo *POST*, riscrivendo il metodo *send()* della midlet e implementando il metodo *doPost()* sulla servlet, lasciando a quest'ultimo il compito di frammentare il messaggio in più parti. Il gioco però, probabilmente, non sarebbe valso la candela, dato che avremmo rischiato di perdere la compatibilità con alcuni terminali meno recenti, che non hanno bene implementato il metodo *Http Post*.

UN CONTACARATTERI

Nello scorso articolo avevamo accennato ad un contatore di caratteri, che ora è ancora più utile, per poter sapere quanti messaggi stiamo per inviare.

Fig. 3: ***La schermata principale, dopo le modifiche. Abbiamo un contacaratteri evoluto e un contatore di messaggi inviati nella giornata attuale***

Vogliamo sapere non solo quanti caratteri abbiamo scritto, ma anche il numero di messaggi che invieremo. Scriviamo una funzione che divide un intero per un altro, restituendoci il resto ed una che ci restituisca il numero di messaggi da inviare:

```
public int resto(int dividendo, int divisore){
  int resto=0;
  while (dividendo > divisore){
    dividendo= dividendo - divisore; }
   resto= dividendo;
   return resto; }
public int quoziente(int dividendo, int divisore){
  int quoziente=1;
  while (dividendo > divisore){
    dividendo= dividendo - divisore;
    quoziente++;
  }
  return quoziente;
}
```

Potremo usare questi due metodi nella classe *ContaCaratteriListener* che implementa *ItemStateListener* e ci aggiorna su tutti i cambiamenti di stato dei vari *Item* contenuti nella *Form* di immissione dati:

```
public class ContaCaratteriListener implements
                                   ItemStateListener{
  StringItem numero;
  TextField testo;
  public ContaCaratteriListener(
                       StringItem numero,TextField testo){
    this.numero=numero;
    this.testo=testo; }
  public void itemStateChanged(Item item){
   int size=testo.size();
   numero.setText(quoziente(size,120)
                                +"/"+resto(size,120));
   }
}
```

E' difficile riuscire a scrivere una midlet a carattere testuale-interattivo con una buona grafica. I vari Item sono molto spartani e poco configurabili per quanto riguarda il look. I più esigenti dovrebbero scriversi i propri item e gestirne manualente il funzionamento. Non è comunque una pratica consigliabile, si perderebbe per esempio la scrittura assistita (T9, iTap, ...)

www.symbian.org
il riferimento ufficiale per sviluppatori symbian.

www.newlc.com
la più grande comunità di sviluppatori symbian.

Il listener deve essere "attaccato" alla form:

```
aForm.setItemStateListener(
                    new ContaCaratteriListener(chars,mex));
```

LO SPLASH-SCREEN INIZIALE

Purtroppo sono poche le finezze grafiche che uno sviluppatore j2me si può permettere in una applicazione basata sull'interazione testuale con l'utente. Sarebbe stato esteticamente più efficace e anche utile, se avessimo potuto visualizzare il messaggio da inviare su più righe, nella sua composizione. Per fare questo, però, avremmo dovuto utilizzare una *TextBox* invece di una *TextField*, che non è derivata da *Item* ma da *Screen* e non può quindi essere visualizzata in una *Form* con altri elementi grafici, ma solo in maniera esclusiva. Non avremmo potuto quindi verificare, durante la scrittura del messaggio, la sua lunghezza, non potendo visualizzare il contatore. Abbiamo quindi preferito la funzionalità all'estetica. Possiamo però azzardare un gustoso splash-screen iniziale, che si visualizza per 5 secondi, per poi lasciare spazio alla form di immissione dati. Deriveremo direttamente da *Canvas*, uno *Screen* molto "primitivo", nel quale dobbiamo gestire direttamente il metodo *paint (Graphics g)* e addirittura catturare la pressione dei tasti attraverso *keyPressed (intkeycode)*. Nell'implementazione del metodo *paint* dobbiamo far disegnare ogni stringa o altro elemento grafico, indicando precisamente il punto di inizio (espresso in coordinate *x,y*) e la sua dimensione. Il tutto risulta quindi piuttosto lento e complicato, ma è l'unico modo per poter sovrapporre immagini e stringhe e per poter scegliere i colori. Per avere uno splash screen, dovremmo impostare un timeout dopo il quale far apparire la schermata successiva. Per farlo istanziamo un *Timer* facendo schedulare un *TimerTask* da eseguire dopo un delay di alcuni secondi:

Fig. 4: ***La stretta connessione fra le classi Form e ContaCaratteriListener. Ad ogni cambiamento di stato di un Item nella Form segue l'esecuzione del metodo itemStateChanged(Item item)***

Fig. 5: ***Lo splash screen. La sua visualizzazione dura 5 secondi. Lasciamo agli amanti della grafica il compito di renderla più accattivante!***

```
public class AboutScreen extends Canvas {
  Image sfondo;
  int width,height;
  public AboutScreen() throws Exception {
     width = getWidth();
     height = getHeight();}
  public class task extends TimerTask{
    public void run(){
      raccogli();
    }
  }
  protected void paint(Graphics g){
    try{
      sfondo=Image.createImage("/sfondo.png");
    }catch (Exception e)
   {
    }
    g.setColor(0xC35BD9);
    g.fillRect(0, 0, width, height);
    g.setColor(0xDEDEDE);
    g.drawImage(sfondo, (width/2)-(
                    sfondo.getWidth()/2), 5, 4 | 0x10);
    g.drawString("SmsSender ver 1.0:", (width/2)-(
                              45), 70, 4 | 0x10);
    g.drawString("Invia fino a 10 sms gratis a", (
                       width/2)-(66), 80, 4 | 0x10);
    g.drawString("tutti i cellulari Vodafone Italia", (
                       width/2)-(66), 90, 4|0x10);
    g.drawString("E' un servizio offerto agli", (
                      width/2)-(62), 110, 4 | 0x10);
    g.drawString("iscritti a www.190.it", (width/2)-(
                             57), 120, 4 | 0x10);
    Timer t= new Timer();
    t.schedule(new task(),5000);
  }
  public void keyPressed(int keyCode){
    raccogli();
 }
}
```

Utilizzeremo questa *Screen* anche come finestra di *"About"*, nel metodo omonimo, rendendo sicuramente più curato l'aspetto della nostra amata creatura:

```
public void about() {
  try{
    AboutScreen ab= new AboutScreen();
    d.setCurrent(ab);
  }
  catch (Exception ee){}
}
```

Ovviamente altre finezze potrete trovarle voi, magari visualizzando nello splash-screen più immagini in sequenza, separate da delle pause (con più *Timer*) in modo da formare una sorta di animazione.

Luca Mattei

Creare GUI Java attraverso il form editor di Visual Basic

Interfacce grafiche da VB a Java

parte seconda

In questa occasione approfondiremo le tematiche della rappresentazione vettoriale in Java e studieremo la struttura dei menu delle interfacce sulla piattaforma di Sun

Il mese scorso abbiamo visto come costruire un programma in grado di convertire un'interfaccia grafica creata tramite l'ambiente Visual Basic, in codice Java. Se avete perso il numero scorso riuscirete comunque a seguire questo articolo perciò non demordete. Creeremo delle funzioni in grado di convertire il codice per creare una barra dei menu, oltre a delle pratiche funzioni per realizzare la grafica di poligoni creata tramite l'oggetto *Shape* di Visual Basic. Funzioni che analizzeremo in questo articolo. Tralasceremo, invece, la conversione degli oggetti *Image* in quanto, nel codice associato a questi ultimi, non viene memorizzato il percorso del file immagine associato. Questo rende inutile una importazione che comporterebbe, comunque, delle modifiche manuali da parte del programmatore.

Fig. 1: **Il programma come si presenta una volta compilato ed avviato**

IL FORMATO DEI FILE VISUAL BASIC

Per chi avesse perso la puntata del mese scorso, riassumiamo brevemente il formato dei file che andremo a trattare. Quando si crea un nuovo form in Visual Basic, il sorgente viene memorizzato in un file con estensione *.frm*. All'interno di questo file viene memorizzato in testa il codice necessario per disegnare e visualizzare il form, comprensivo di tutti gli oggetti associati quali: pulsanti, caselle di testo, etc… Il linguaggio utilizzato da Visual Basic per la creazione degli oggetti grafici è simile al Pascal: il codice di ogni oggetto viene racchiuso tra le istruzioni *<Begin.VB><nome oggetto> …. <END>*. Tra questi due costrutti, poi, vengono indicate le varie proprietà dell'oggetto in questione tramite una organizzazione *nome proprietà = valore*. Le varie proprietà vengono specificate una per ogni riga ed hanno una spaziatura fissa nel senso che, a partire dalla colonna in cui viene specificata la clausola *Begin.VB*, il nome della proprietà viene indicato nella colonna 4 ed occupa al massimo 16 caratteri dopodichè viene indicato il carattere "=", e alla colonna 21 inizia la specifica del valore della proprietà. Oltre a questo, va aggiunto che un oggetto potrebbe essere figlio di un altro oggetto nel senso che, ad esempio, un pulsante potrebbe essere disegnato all'interno di un altro pulsante o, come spesso accade, più oggetti potrebbero essere disegnati all'interno di un oggetto *Frame*. Questa gerarchia di oggetti viene risolta da Visual Basic inglobando all'interno del codice *Begin VB. … End* un altro blocco di codice appartenente ai vari oggetti figli.

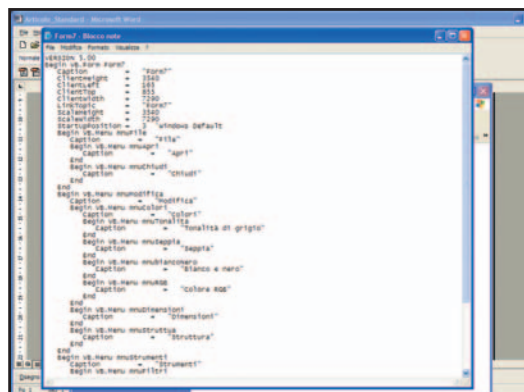



Fig. 3: **Dallo scorcio è chiaramente visibile la gerarchia padre/figlio dei vari oggetti esposti**

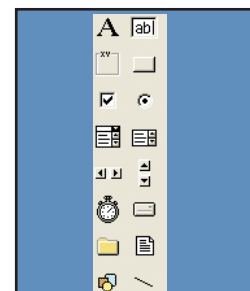



Fig. 2: **Gli oggetti di default utilizzabili all'interno dell'ambiente Visual Basic**

CREARE I MENU

Per creare un menu in Java bisogna prima creare un oggetto di tipo *JMenuBar*, che rappresenta la barra dei menu. All'interno di questo vanno inserite le varie voci che sono rappresentate da oggetti di tipo *JMenu*. Infine, ogni *JMenu* può contenere, a sua volta, altri *JMenu* (nel caso di sottomenu) oppure oggetti di tipo *JMenuItem* che rappresentano le voci effettivamente cliccabili del menu.

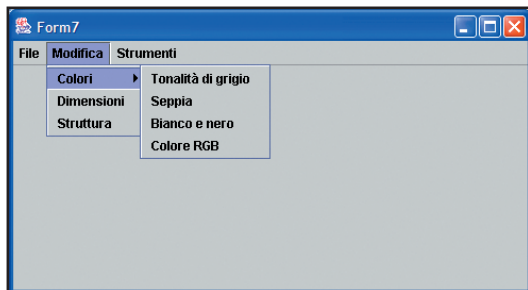



Fig. 4: ***Un'immagine che mostra un menu con i relativi sottomenu***

Detto questo, come prima cosa creiamo una variabile statica e pubblica all'interno del file *SwingCreator .java*

```
public static boolean barraMenu;
```

Tale variabile, all'interno del costruttore, sarà impostata a *false*. Ogni qualvolta nella lettura del file sorgente *.frm* verrà incontrata l'istruzione *Begin.VB Menu*, questa variabile verrà impostata a *true*, ad indicare che dovranno essere inserite le istruzioni atte a creare la barra dei menu in Java.

Le altre istruzioni riguardanti la creazione dei menu, da includere in questo file, sono definite all'interno del metodo *inserisciOggetto*. Tale metodo è utilizzato in maniera ricorsiva per scandire una riga alla volta il file *.frm*, individuare l'inizio di un oggetto, memorizzare le sue proprietà e verificare se possiede figli oppure no. Il codice aggiunto al metodo è il seguente:

```
...
int indiceTemp;
...
indiceTemp = indice;
...
if(index != -1) {
   oggetto[indiceTemp].setGenitore();
```

Dichiariamo una variabile di tipo *int* che servirà a memorizzare il valore temporaneo dell'indice. Una volta entrati nel ciclo *do/while*, nel caso in cui esista un oggetto figlio, all'oggetto padre viene settata la proprietà genitore su *true*, tramite il metodo *setGenitore*, utilizzando come indice dell'oggetto il valore di *indiceTemp*. Questo nuovo metodo è definito all'interno del file *Oggetto.java*, file che contiene le definizioni degli oggetti creati. Ricordo che, ogni qualvolta si riscontra una riga nel file *.frm*, contenente la definizione *Begin.VB*, viene creato un nuovo oggetto che conterrà come proprietà quelle definite nel sorgente Visual Basic. All'interno di questa classe dobbiamo definire i seguenti due metodi:

```
public void setGenitore() {
   this.genitore = true; }
public boolean getGenitore() {
   return genitore;
}
```

Tali metodi servono rispettivamente a settare la variabile genitore a *true* ed a restituirci il valore di tale proprietà. Questa proprietà sarà utilizzata dalla *classe ScriviFile* nel momento in cui vorremo scrivere il codice necessario a generare il menu in Java. Infatti, se la proprietà genitore è impostata a *true*, avremo la sicurezza che l'oggetto da creare è di tipo *JMenu;* in caso contrario significa che ci troveremo di fronte alla voce di menu realmente cliccabile quindi l'oggetto da creare sarà di tipo *JMenuItem*. Il codice che si trova all'interno del metodo *dichiarazioni* e del metodo *inserisciOggetto;* il primo si occupa di creare il codice riguardante le dichiarazioni degli oggetti in Java, il secondo di tutti gli oggetti rilevati e, nel caso corrispondano con quelli riconosciuti dal programma, di creare il codice Java. Vediamo come è costruito il codice all'interno del metodo *dichiarazioni*:

```
else if(obj[i].getTipo().equals("Menu")) {
   if(obj[i].getGenitore())
   pw.println("  private JMenu " + obj[i].getNome() + ";");
   else
      pw.println("  private JMenuItem " +
                                    obj[i].getNome() + ";");
      SwingCreator.barraMenu = true;
}
```

Viene letta la proprietà *tipo* tramite il metodo *getTipo()*. Se la proprietà equivale a *Menu* viene verificata la proprietà genitore. A seconda del valore restituito verrà creato un oggetto di tipo *JMenu* oppure *JMenuItem*. Infine, viene impostata la variabile statica *barraMenu* su *true*. Il codice presente nel metodo *inserisciOggetto* è praticamente equivalente.

GRAFICA VETTORIALE IN JAVA

Passiamo adesso a vedere come convertire il codice proveniente dal disegno di figure geometriche quali rettangoli, quadrati, cerchi ed ellissi tramite gli oggetti *Shape* di Visual Basic. Per quanto riguarda il codice VB, queste figure vengono create nella stessa

ANTEPRIMA

Per poter visualizzare l'anteprima a video dovrete assicurarvi di copiare correttamente i file a corredo dei sorgenti ap.bat nella stessa cartella in cui è contenuto il programma compilato.

maniera di tutti gli altri oggetti. Quello che cambia, invece, è il codice Java. Infatti, per poter disegnare figure geometriche in Java abbiamo bisogno, innanzitutto, di un oggetto di tipo *JPanel* che fungerà da lavagna; dopodichè, dovremo ridefinire il metodo *paintComponent* di questa classe al cui interno verranno implementate le istruzioni di disegno. Il procedimento descritto si può schematizzare in quattro punti fondamentali:

1. Creazione di una classe che estende *JPanel*
2. Ridefinizione del metodo *paintComponent* all'interno del quale saranno presenti le istruzioni di disegno
3. Creazione di un oggetto della classe sopra descritta
4. Aggiunta di questo oggetto al *Container*

Per fare ciò all'interno del nostro programma dichiariamo, inanzitutto, due variabili booleane di tipo statico all'interno della classe *SwingFrame:*

```
public static boolean disegno;
public static boolean geometrie;
```

La variabile *disegno* servirà a registrare il valore dello stato dell'oggetto *JCheckBox chkDisegno.* Nel caso in cui la voce sia selezionata, saremo in presenza di un form contenente delle figure geometriche. Vedremo più avanti come utilizzare tale variabile per costruire il codice che realizzerà il pannello su cui disegnare. La variabile *geometrie*, invece, verrà settata a *true* all'interno del metodo *inserisciOggetto* della classe *ScriviFile* ogniqualvolta si riscontra un oggetto di tipo *Shape.*
Quando l'importazione sarà conclusa, se questa variabile sarà impostata a *true* e la variabile *disegno* sarà impostata a *false* avremo la certezza che non sono state importate le istruzioni di disegno, in quanto nessun pannello grafico è stato creato. Analizziamo ora il caso in cui avremo selezionata correttamente l'opzione *Includi disegno figure geometriche.* La prima cosa che viene fatta è dichiarare una variabile di tipo *JPanel* nel metodo intestazione della classe *ScriviFile:*

```
if(SwingCreator.disegno)
   pw.println("  private Disegno pannelloDisegno;");
```

La classe *Disegno,* come vedremo, sarà una nuova classe che estenderà *JPanel.* A questo punto, all'interno del metodo dichiarazioni andremo a istanziare un oggetto da tale classe e lo aggiungeremo al *Container:*

```
if(SwingCreator.disegno) { // Inserisce anche la
                         lavagna su cui disegnare
   pw.println(spazio + "pannelloDisegno = new Disegno();");
```

```
   pw.println(spazio + "pannelloDisegno.setLayout(null);");
   pw.println(spazio + "pannelloDisegno.setBounds(
                   0,0," + obj[0].getClientWidth() + "," +
                        obj[0].getClientHeight() + ");");
   pw.println(spazio + "c.add(pannelloDisegno);");
   pw.println("");
}
```

L'oggetto istanziato avrà le medesime dimensioni della finestra, in quanto sarà completamente sovrapposto al *Container.* C'è da notare che anche gli oggetti quali pulsanti, caselle di testo, etc.. verranno aggiunte a tale pannello. Infine, nel metodo *pieDiPagina,* nel caso in cui la variabile disegno valga *true* verrà invocato il metodo *aggiungiGrafica.* All'interno di questo metodo, dopo aver inserito le istruzioni necessarie a creare la classe, viene realizzato un ciclo *for,* all'interno del quale sono tale ciclo, vengono analizzati tutti gli oggetti registrati.
Ogni qualvolta viene individuato un oggetto di tipo *Shape* si passa al controllo della proprietà *shape.* I valori che può assumere tale proprietà sono riportate in **Tabella 1**. Riportare tutto il codice del metodo non è possibile per motivi di spazio; riporteremo, come esempio, solamente il caso in cui la variabile *shape* assuma il valore 1:

Valore	Figura da disegnare
0	Rettangolo
1	Quadrato
2	Ovale
3	Cerchio
4	Rettangolo con angoli arrotondati
5	Quadrato con angoli arrotondati

Tabella 1: Valori che può assumere la proprietà Shape

```
case 1: // Quadrato
width = Integer.parseInt(obj[i].getWidth());
height = Integer.parseInt(obj[i].getHeight());
left = Integer.parseInt(obj[i].getLeft());
top = Integer.parseInt(obj[i].getTop());
if(width > height)
{
   left = left + (width / 2) - (height / 2);
   width = height;
}
else
{
   top = top + (height / 2) - (width / 2);
   height = width;
}
if(obj[i].getBackStyle() == 1)
{
   pw.println(spazio + "   g.setColor(new Color("
                          + obj[i].getBackColor() + "));");
   pw.println(spazio + " g.fillRect(" + left + ","+ top + ","
                          + width + "," + height + ");");
}
pw.println(spazio + "   g.setColor(new Color("
                          + obj[i].getBorderColor() + "));");
pw.println(spazio + " g.drawRect(" + left + ","+ top + ","
                          + width + ","+ height + ");");
```

FILE BATCH

I file batch sono in realtà documenti di tipo testuale al cui interno sono contenute istruzioni per l'ambiente DOS. Queste istruzioni vengono lette in sequenza ed interpretate nel momento in cui il file viene lanciato

Abbiamo considerato come esempio il quadrato, in quanto, i valori di larghezza e di altezza potrebbero essere diversi. Questo perché l'oggetto *Shape* di VB traccia, in realtà, un'area rettangolare che viene riempita a seconda di quello che si sceglie di disegnare. In questo caso, per ottenere le dimensioni giuste, bisogna verificare quale delle due dimensioni è la più piccola: questa rappresenterà la dimensione del lato (nel caso del cerchio sarà la dimensione del diametro).

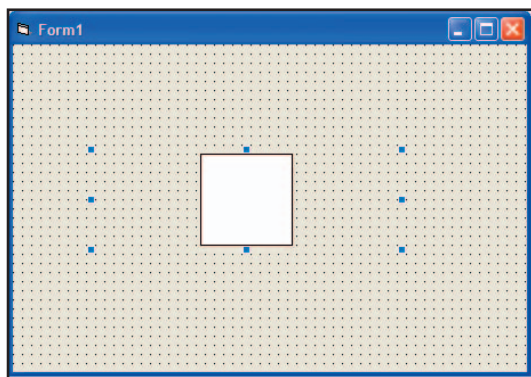



Fig. 5: *L'area selezionata è quella contenuta all'interno dei vertici contrassegnati da piccoli quadratini blu*

A questo punto, va verificata la proprietà *backStyle*. Tale proprietà, se ha valore 1, segnala che il bordo esterno potrebbe essere di colore differente dal colore utilizzato per disegnare l'intera figura. Per risolvere questo problema basta disegnare prima l'intera figura, utilizzando come metodo *fillRect*, il quale, dopo aver tracciato il perimetro esterno lo riempie con il colore scelto, dopodichè lo stesso perimetro lo si disegnerà utilizzando, questa volta, il metodo *drawRect*. Per disegnare i rettangoli ed i quadrati con gli angoli arrotondati, i metodi utilizzati sono invece *fillRoundRect* e *drawRoundRect* ai quali passiamo due argomenti aggiuntivi uguali a 10.

AVVIARE PROCESSI DA JAVA

Prima di concludere diamo un breve sguardo alla funzione utilizzata per avviare l'anteprima a video dell'interfaccia creata.
Alla fine del metodo *iniziaConversione* della classe *SwingCreator* viene analizzato il contenuto della variabile *anteprima*. Se tale variabile vale 1 significa che si è scelto di visionare l'anteprima a video), viene creato un'istanza di *Anteprima*.
Il codice della classe è il seguente:

```
import java.io.*;
import javax.swing.JOptionPane;
public class Anteprima {
  private PrintWriter pw;
  public Anteprima(String nomeClasse) {
    String spazio = "              ";
     try {
        pw = new PrintWriter(
        new BufferedWriter(
        new FileWriter(
        new File("C:\\Test.java"))));
        pw.println("public class Test {");
```

```
            pw.println(" public static void main(String[]
                                                args) {");
            pw.println(spazio + nomeClasse + "app = new"
                                   + nomeClasse + "();");
            pw.println(spazio + "app.show();");
               pw.println("   }");
               pw.println("}");
               pw.close();
               Process p = Runtime.getRuntime(
                                    ).exec("ap.bat");
        }
        catch(IOException e)
        {
             JOptionPane.showMessageDialog(null,
             "Attenzione! Si sono verificati degli errori
                      nella creazione dell'anteprima.",
             "Swing Creator",
             JOptionPane.ERROR_MESSAGE);
             System.out.println(e.getMessage());
      }
   }
}
```

All'interno del costruttore viene realizzato un file di nome *Test.java* che conterrà il metodo *main* incaricato di caricare la classe creata in precedenza dalla conversione del file Visual Basic. Dopodichè viene lanciato un file *batch* contenente le seguenti istruzioni:

```
C:
cd\
javac Test.java
java Test
```

Il file *batch*, in pratica, richiama l'ambiente DOS, compila il programma e lo avvia. Per avviare il file creiamo un oggetto di tipo *Process* associandogli un riferimento di tipo *Runtime*. Il riferimento ci viene restituito dal metodo *getRuntime()*.
A questo punto basta lanciare il metodo *exec* passandogli come parametro il nome del programma da mandare in esecuzione.

CONCLUSIONI

Oltre al codice presentato all'interno del programma, sono state implementate nuove istruzioni.
Vi consiglio di guardarle, anche se a questo punto dovreste essere in grado di continuare da soli nell'inserimento di nuove funzioni.
Il programma è totalmente funzionante ed anche questo mese, insieme ai sorgenti, vengono forniti degli esempi di form in VB per testare le qualità e le potenzialità di *SwingCreator*.
Alla prossima.

Alessandro Baldini

CONTROLLI VB

Esistono numerosi controlli in Visual Basic oltre a quelli presenti nella barra standard, per accedere ai quali bisogna portarsi sulla voce di menu *Progetto* e scegliere la voce *Componenti...* Dalla finestra che appare si potrà così scegliere il componente preferito selezionandolo con una spunta.

Ritorno al futuro: l'output formattato

Le novità di Java 1.5

parte terza

Java 1.5 presenta un sistema per la formattazione dell'output. Questa caratteristica, già nota ai programmatori C/C++, C# e Visual Basic, sarà l'argomento di questo articolo

Analizzando le "nuove" caratteristiche relative alla formattazione dell'output, si ha la sensazione di tornare indietro di qualche decennio. Ritroveremo vecchi concetti già noti ai tempi del C, come la funzione *printf* ed i codici di formattazione *%s, %d, %f*, ecc... Java 1.5, nel 2004, introduce delle "novità" già presenti in C nel 1970: ah, il progresso!

OUTPUT FORMATTATO

La classe *PrintStream* di Java 1.5 implementa il metodo *format* che scrive sullo stream di output dati formattati secondo un preciso criterio passato in input. Se eseguiamo il seguente frammento di codice:

```
int e = 29;
System.out.format("Io ho %d anni\n",e);
```

otterremo in output:

Io ho 29 anni

Cerchiamo di capire cosa sia successo:
il metodo *format* riceve in input una stringa ed un valore intero. La stringa è chiamata stringa di formattazione e contiene il simbolo speciale *%d*. Il metodo la scriverà sullo stream di output sostituendo il simbolo speciale con il valore del secondo parametro, cioè la variabile *e*.
Consideriamo ora quest'altro esempio:

```
int e = 29;
String nome = "Mario";
System.out.format(
                "Io sono %s ed ho %d anni\n",nome,e);
```

Adesso la stringa di formattazione contiene due simboli speciali: *%s* e *%d*. L'output sarà quindi la stringa stessa nella quale al simbolo *%s* si sostituiscono il valore della stringa *nome* e al simbolo *%d* il valore della variabile *e*:

Io sono Mario ed ho 29 anni

Come avrete notato, sono stati usati simboli diversi per la stringa *(%s)* e per l'intero *(%d)*. In questo modo, il metodo sarà "informato" a priori sul tipo di variabile che seguirà. Come vedremo nei prossimi paragrafi, esistono numerosi simboli speciali, tutti preceduti da %, associati a diversi tipi di dato. Per ragioni prettamente storiche, la classe *PrintStream* implementa anche il metodo *printf*, che ha la stessa identica funzionalità del metodo *format*.
L'esempio precedente può quindi anche essere riscritto così:

```
int e = 29;
String nome = "Mario";
System.out.printf(
                "Io sono %s ed ho %d anni\n",nome,e);
```

Il metodo *printf* non fa altro che richiamare il metodo *format*. Anche la classe *String* implementa il metodo *format*. È un metodo statico che funziona esattamente come il suo omonimo di *PrintStream*, ma invece di scrivere l'output su uno *stream*, lo restituisce come stringa. Il seguente è un esempio d'utilizzo:

```
int e = 29;
String nome = "Mario";
String s = String.format(
                "Io sono %s ed ho %d anni \n",nome,e);
System.out.println(s);
```

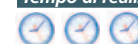
Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
C# (basi), Web Services, base in Java, base in Visual Basic

Software
SDK 1.3 o superiore

Impegno

Tempo di realizzazione

Come i metodi *format* e *printf* di *PrintStream* sono gli equivalenti della funzione *C printf*, il metodo *format* di *String* è l'equivalente della funzione *sprintf*.
L'arguto lettore avrà di certo notato che il metodo *format* può avere in input una lista variabile di argomenti.

SIMBOLI DI CONVERSIONE PER INTERI

Nelle stringhe di formattazione esistono dei simboli speciali, denominati *simboli di conversione*, per ogni categoria di tipo di dato.

Simbolo	Descrizione
%d	Intero in notazione decimale
%o	Intero in notazione ottale
%x	Intero in notazione esadecimale

Tabella 1: Simboli di conversione per variabili intere

Nella **Tabella 1**, riportiamo i simboli per le variabili intere. Per variabile intera, si intende quella che rappresenta numeri interi, come *int*, *short*, *long*, ecc...
Lo stesso valore numerico apparirà in un diverso formato secondo il simbolo utilizzato. Infatti, se eseguiamo il seguente frammento di codice:

```
int e = 29;
System.out.format("Io ho %d anni (decimale)\n",e);
System.out.format("Io ho %o anni (ottale)\n",e);
System.out.format("Io ho %x anni (esadecimale)\n",e);
```

otterremo:

Io ho 29 anni (decimale)
Io ho 35 anni (ottale)
Io ho 1d anni (esadecimale)

Simbolo	Descrizione
%f	Numero decimale
%e	Numero decimale in notazione scientifica
%g	Numero decimale in notazione scientifica (per esponenti molto elevati o molto piccoli)
%a	Numero esadecimale in notazione virgola mobile

Tabella 2: Simboli di conversione per i numeri decimali

Nel primo caso il numero 29 sarà espresso in notazione *decimale(29)*, nel secondo in *ottale(35)* e nel terzo in *esadecimale(1d)*. I simboli di conversione per gli interi, dopo % e prima della lettera, possono assumere un valore numerico. Esso indica la lunghezza minima (in caratteri) che il valore sostituito deve avere. Nei casi in cui tale lunghezza non viene raggiunta, verranno aggiunti spazi vuoti in testa (giustificazione a destra). Consideriamo a tale proposito il seguente esempio:

```
System.out.format("%2d %2d %2d\n",3,13,2);
System.out.format("%2d %2d %2d\n",44,4,8);
```

Il simbolo definito nella stringa di formattazione è *%2d*, ciò significa che il valore sostituito dovrà essere lungo almeno due caratteri. Per i numeri 3, 2, 4 e 8 sarà quindi aggiunto uno spazio addizionale in testa. L'output sarà il seguente:

```
 3 13  2
44  4  8
```

Questo modo di operare è molto utile quando si desidera ottenere dei valori ben incolonnati.
È possibile anche ottenere la giustificazione a sinistra; per far ciò, basta inserire un valore negativo:

```
System.out.format("%-2d %-2d %-2d\n",3,13,2);
System.out.format("%-2d %-2d %-2d\n",44,4,8);
```

In questo caso è stato usato *-2*.
L'output sarà il seguente:

```
3  13 2
44 4  8
```

Noterete che i numeri sono giustificati a sinistra. Come già abbiamo detto, i simboli di conversione per gli interi funzionano non solo con int ma anche con altri tipi di dato intero come *short* e *long*. Ecco un esempio:

```
//short
short sh = 18;
System.out.format("sh = %d\n",sh);
//long
long l = 9000000000L;
System.out.format("l = %d\n",l);
```

I programmatori C avranno di certo notato che in Java per il tipo *long* viene utilizzato *%d* e non *%ld*.

SIMBOLI DI CONVERSIONE PER VALORI DECIMALI

Nella **Tabella 2** sono riportati i simboli di conversione per i numeri decimali. Questi simboli possono essere utilizzati con variabili decimali (*float*, *dauble*, ecc...) in maniera simile a quella vista per le variabili intere. Di seguito ne riportiamo un esempio:

```
double x = Math.sqrt(2);
System.out.format("La radice di due è: %e\n", x);
System.out.format("La radice di due è: %f\n", x);
System.out.format("La radice di due è: %g\n", x);
System.out.format("La radice di due è: %a\n", x);
```

L'output del codice appena visto sarà:

La radice di due è: 1.414214e+00
La radice di due è: 1.414214
La radice di due è: 1.414214
La radice di due è: 0x1.6a09e667f3bcdp0

All'interno del simbolo di conversione è possibile specificare anche il numero massimo di cifre intere e decimali che devono essere visualizzate. La sintassi completa è infatti la seguente:

```
%[ampiezza][.precisione]<f | g | e | a>
```

Nell'esempio seguente verrà visualizzato un numero decimale con quattro cifre intere e con al massimo due decimali:

```
System.out.format("%4.2f", 1024.879178);
```

Il risultato è quindi:

1024.88

invece di:

1024.879178

poiché il numero, come specificato dal simbolo di conversione, è stato arrotondato alla seconda cifra decimale.

SIMBOLI DI CONVERSIONE PER STRINGHE E CARATTERI

Il simbolo di conversione per le stringhe *(%s)* lo abbiamo già visto nei paragrafi precedenti. Esso viene inserito nella stringa di formattazione per specificare variabili di tipo *String*, come mostrato nel seguente esempio:

```
String str = "Ciao";
System.out.format("%s, come stai?\n", str);
```

Anche per *%s* è possibile specificare la giustificazione a destra o a sinistra. Eseguendo il codice che segue:

```
// Giustificazione a destra
System.out.format(
        "%5s %5s %5s\n","ciao","casa","prato");
System.out.format("%5s %5s %5s\n\n","tre","si","a");
// Giustificazione a sinsitra
System.out.format(
        "%-5s %-5s %-5s\n","ciao","casa","prato");
System.out.format("%-5s %-5s %-5s\n","tre","si","a");
```

si otterrà:

ciao casa prato
tre si a

ciao casa prato
tre si a

Nel primo caso, la giustificazione sarà a destra. Nel secondo caso invece, utilizzando il valore numerico negativo –5, le stringhe saranno giustificate a sinistra. Il simbolo di conversione per i caratteri (tipo *char*) è *%c*. Di seguito ne riportiamo un esempio:

```
char ch = 'A';
System.out.format("ch = %c\n",ch);
```

Utilizzando *%c* con un valore numerico, in output otterremo il carattere il cui codice ASCII corrisponde al valore numerico passato in input:

```
int code = 65;
System.out.format("ch = %c\n",code);
```

L'output sarà quindi:

ch = A

65 è infatti il codice ascii del carattere *'A'*.

FORMATTAZIONE DI DATE E ORARI

Il simbolo di formattazione *$t* si applica a variabili rappresentanti date o orari, ovvero di tipo *long, Calendar* o *Date*. Il funzionamento di *$t* è leggermente diverso da quello dei simboli visti in precedenza. Date e orari non sono tipi semplici, ma composti. Da una data si possono estrarre informazioni quali il giorno, il mese, l'anno, il giorno della settimana, ecc... Ciò accade anche per gli orari; possiamo considerare l'ora, i minuti, i secondi e perfino i millisecondi. Data tale complessità, è comprensibile aspettarsi che il simbolo *$t* sia più complicato dei suoi colleghi. Infatti, associato ad esso compare sempre un secondo simbolo che indica quale "sotto" informazione deve essere considerata. Così *$td* sarà il giorno del mese, *$tm* il mese, *$ty* l'anno, e così via.
Per utilizzare il simbolo *$t* in una stringa di formattazione bisogna specificare inoltre il numero di parametro corrispondente della lista degli argomenti. Quindi, *%1$tm* significa: "preleva il mese dal primo parametro presente nella lista dei parametri". È obbligatorio che tale parametro sia di tipo compatibile con il simbolo di conversione *$t*, deve quindi essere una variabile di tipo *long, Date* o *Calendar*.

NOTA

QUANTO È COMPATIBILE IL JDK 1.5 CON I PRECEDENTI?

In teoria dalla versione 1.4.2 in su i binari dovrebbero essere compatibili. Le poche incompatibilità sono elencate all'indirizzo http://java.sun.com/j2se/1.5.0/compatibility.html#incompatibilities

SUL WEB

Sun Microsystems, JavaTM 2 SDK, Standard Edition, Version 1.5.0 - Summary of New Features and Enhancements.
http://java.sun.com/j2se/1.5.0/docs/relnotes/features.html

Calvin Austin, J2SE 1.5 in a Nutshell.
http://java.sun.com/developer/technicalArticles/releases/j2se15/

Sun Microsystems, J2SE 1.5 Documentation, Class Formatter.
http://java.sun.com/j2se/1.5.0/docs/api/java/util/Formatter.html

Sun Microsystems, JSR 201: Extending the JavaTM Programming Language with Enumerations, Autoboxing, Enhanced for loops and Static Import.
http://jcp.org/en/jsr/detail?id=201

Nell'esempio che segue verrà visualizzata una data nel formato *gg/mm/aaaa:*

```
System.out.format(
  "Data: %1$td/%1$tm/%1$tY\n", date);
```

Nella **Tabella 3**, riportiamo i suffissi previsti per le date.

Suffisso	Descrizione	Esempi
B	Nome del mese	*Gennaio*
b	Nome del mese (abbreviaz.)	*Gen*
h	Come b	
A	Giorno della settimana	*Domenica*
a	Giorno della settimana (abbreviaz.)	*Dom*
C	Anno diviso per 100 (due cifre)	*98, 99, 00, 01*
Y	Anno	*1989*
y	Anno (due cifre)	*89*
j	Giorno dell'anno	*001-366*
m	Mese	*01-12*
d	Giorno del mese (due cifre)	*01-31*
e	Giorno del mese	*1-31*

Tabella 3: Suffissi per le date

Nel seguente esempio, la data 5/6/1989 verrà visualizzata in diversi formati:

```
Calendar c =
  new GregorianCalendar(1989, Calendar.JUNE, 5);
// 05/06/1989
System.out.format(
                        "Data: %1$td/%1$tm/%1$tY\n", c);
// 5-06-89
System.out.format(
                        "Data: %1$te-%1$tm-%1$ty\n", c);
// 05 giugno 1989
System.out.format(
                       "Data: %1$td %1$tB %1$tY\n", c);
// lunedì, 5 giugno 1989
System.out.format(
               "Data: %1$tA, %1$te %1$tB %1$tY\n", c);
// 5 giu 89
System.out.format("Data: %1$te %1$tb %1$ty\n", c);
```

l'output sarà:

Data: 05/06/1989
Data: 5-06-89
Data: 05 giugno 1989
Data: lunedì, 5 giugno 1989
Data: 5 giu 89

I suffissi per gli orari sono riportati in **Tabella 4**. Nell'esempio che segue, l'ora corrente verrà visualizzata secondo diversi formati:

```
Calendar time =
  Calendar.getInstance(); //ora attuale
// 15:19:10
System.out.format(
                 "Ora: %1$tH:%1$tM:%1$tS\n", time);
// 15.19
System.out.format("Ora: %1$tH.%1$tM\n", time);
// 15:19:10.456000000
System.out.format(
       "Ora: %1$tH:%1$tM:%1$tS.%1$tN\n", time);
// 3.19PM
System.out.format(
                 "Ora: %1$tl.%1$tM%1$tp\n", time);
```

Avremo quindi in output:

Ora: 15:19:10
Ora: 15.19
Ora: 15:19:10.456000000
Ora: 3:19PM

Negli esempi precedenti abbiamo usato *%1* come numero di parametro. La numerazione dei parametri è molto importante. Essa è stata introdotta per evitare ambiguità nell'uso di diversi simboli *$t* all'interno della stessa stringa di formattazione o in concomitanza con altri simboli di conversione.
Nell'esempio che segue verrà utilizzato *%2* come numero di parametro poiché il primo sarà una stringa:

```
String str = "Data";
System.out.format(
                 "%s: %2$td/%2$tm/%2$tY\n", str, c);
```

Suffisso	Descrizione	Esempi
H	Ora (24 ore, due cifre)	00-23
I	Ora (12 ore, due cifre)	01-12
k	Ora (24 ore)	0-23
l	Ora (12 ore)	1-12
M	Minuto (due cifre)	00-59
S	Secondi	00-60 (*) 60 è un valore speciale (leap)
L	Millisecondi	000-999
N	Nanosecondi	000000000-999999999
p	indicatore mattina/pomeriggio (maiuscolo)	AM, PM
P	indicatore mattina/pomeriggio (minuscolo)	am, pm
z	Formato RFC 822	Sat Apr 03 10:27:00 GMT 2004
Z	Abbreviazione per Time/Zone	
s	Secondi trascorsi dal 1/1/1970	
E	Millisecondi trascorsi dal 1/1/1970	

Tabella 4: Suffissi per gli orari

CONCLUSIONI

Beh, anche se non ce molto di nuovo in questa novità di Java 1.5, sicuramente la possibilità di formattare l'output risulterà utile in numerosi contesti.

Giuseppe Naccarato

La Borsa sotto controllo

parte terza

In questo articolo impareremo come eseguire operazioni sui database, diremo qualcosa sulla persistenza dei dati, concluderemo l'applicazione Stock Spy iniziata nei numeri scorsi

CD WEB
StockSpy_3.zip
cdrom.ioprogrammo.it

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
Elementi di Java

Software
J2SE 1.4.1 o superiore

Impegno

Tempo di realizzazione

Nei numeri precedenti di ioProgrammo abbiamo costruito un'applicazione 100% Java che preleva le informazioni sull'andamento della Borsa direttamente da Internet e le mostra a video. In questo numero vedremo come queste informazioni possano utilmente essere salvate in un database per successive elaborazioni.
Se non avete seguito fin qui la serie, imparerete comunque come Java possa facilmente manipolare base di dati.

DI COSA ABBIAMO BISOGNO

Prima di tutto di un database. Negli esempi proposti verrà utilizzato il database Microsoft Access. Nel CD allegato alla rivista, oltre al codice sorgente dell'applicazione completa, è presente anche il file di database *data.mdb*.
Sebbene sia stato utilizzato Access, sarà possibile far funzionare l'applicazione con qualsiasi database relazionale dotato di un driver JDBC.
Per semplicità, la nostra base di dati sarà costituita da una singola tabella, denominata *STOCK_QUOTE*. Essa conterrà i campi esposti in **Tabella 1**.

Nome	Tipo	Descrizione
stock	*Testo*	Nome dell'azione
value	*Numerico*	Valore dell'azione
time	*Testo*	Data e orario dell'ultima quotazione (Formato: AAAA/MM/GG HH:MM)
difference	*Numerico*	Variazione rispetto al prezzo di riferimento del giorno precedente

Tabella 1: I campi della tabella STOCK_QUOTE

ACCESSO AI DATI MEDIANTE JDBC

L'accesso al database avverrà attraverso l'interfaccia standard JDBC. Il nome del driver e l'URL che identifica il database saranno specificate all'interno del file di proprietà *stockspy.properties*. Nel nostro caso, il driver sarà il bridge JDBC-ODBC, l'URL invece localizzerà il DSN *STOCKDATA* (vedi riquadro *Data Source Name*). Le due proprietà saranno quindi:

```
# stockspy.properties
jdbc.driver=sun.jdbc.odbc.JdbcOdbcDriver
jdbc.url=jdbc:odbc:STOCKDATA
```

L'accesso ai dati è delegato alla classe *StockDB* che è composta da tre metodi statici fondamentali: *init*, *save* e *query*.

INIZIAMO

Il metodo *init*, che sarà eseguito una sola volta, effettua la connessione alla fonte dei dati e memorizza l'oggetto *Connection* risultante.
Questa è l'implementazione:

```
import java.util.*;
import java.io.*;
import java.sql.*;
public class StockDB {
  public static Connection con;
  public static void init() throws Exception {
    Properties properties = new Properties();
    properties.load(new FileInputStream(
                                "stockspy.properties"));
    String driver = properties.getProperty("jdbc.driver");
    String url = properties.getProperty("jdbc.url");
    Class.forName(driver).newInstance();
    con = DriverManager.getConnection(url);
    con.setAutoCommit(false);
  }
...
```

Il metodo legge le proprietà *jdbc.driver* e *jdbc.url* dal file *stockspy.properties* e ne memorizza i valori nelle variabili *driver* e *url*, che saranno successivamente usate per stabilire la connessione. Quest'ultima operazione è effettuata mediante l'invocazione del metodo *getConnection* di *DriverManager*. La connessione attiva è quindi rappresentata dall'oggetto *con*. L'ultima operazione è disattivare la modalità *auto-commit;* ciò è necessario per avvalersi delle transazioni.

SALVIAMO I DATI

Il metodo *save* ha il compito di salvare le quotazioni nella tabella *STOCK_QUOTE*. Come ricorderete, le quotazioni sono memorizzate in oggetti appartenenti alla classe *StockQuote*. Il metodo *save* memorizzerà sul database i dati provenienti dall'istanza di *StockQuote* in input:

```
public static void save(StockQuote stockQuote)
throws Exception {
 String sql =
   "INSERT INTO STOCK_QUOTE VALUES (" +
   /* stock */
   "'" + stockQuote.name + "'," +
   /* value */
   + stockQuote.value + "," +
   /* time */
   "'" + formatDateTime(stockQuote.date) + "'," +
   /* difference */
   + stockQuote.difference +
   ")";
 Statement st = con.createStatement();
 st.executeUpdate(sql);
 con.commit();
}
```

Come si può notare, il metodo costruisce il comando SQL *INSERT* utilizzando il valore degli attributi relativi all'oggetto *stockQuote* in input. Il metodo *formatDateTime*, implementato all'interno della classe *StockDB*, converte una data dal formato *java.util.Calendar* alla stringa accettata dalla nostra tabella *STOCK_QUOTE*, cioè in formato *"AAAA/MM/GGHH:MM"*. Il comando viene inoltrato al database mediante il metodo *executeUpdate* dell'oggetto *Statement*.

INTERROGHIAMO IL DATABASE

Il metodo *query* è il più complesso dei tre. Esso svolge in pratica due funzionalità: richiesta di quotazioni *multiday* e richiesta di quotazioni *intraday;* nel primo caso restituirà una singola quotazione per ogni giorno (l'ultima del giorno), nel secondo invece tutte le quotazioni di un singolo giorno. Chiariamo il concetto con un esempio. Supponiamo che per l'azione ENI il nostro database abbia i seguenti valori:

Valore	Tempo
16.40	*2004/07/06 12:00*
16.43	*2004/07/06 17:00*
16.61	*2004/07/07 09:00*
16.59	*2004/07/07 12:00*
6.63	*2004/07/07 17:00*

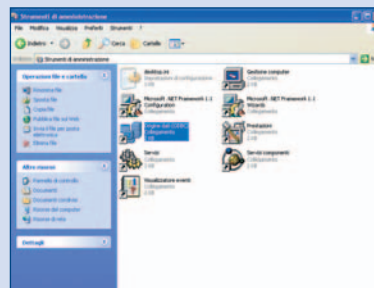
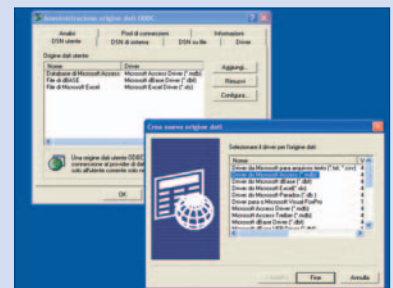
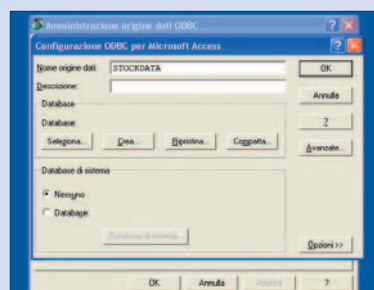
Se siamo interessati ad una consultazione *multiday*, allora il metodo restituirà:

Valore	Tempo
16.43	*2004/07/06 17:00*
16.63	*2004/07/07 17:00*

ovvero le quotazioni con orario più alto relative al 6 e 7 Luglio. Se invece siamo interessati ad una consultazione *intraday* per il giorno 6 Luglio, co-

• **JAVA DATABASE E PROGRAMMAZIONE CLIENT/SERVER**
G. Naccarato
(Apogeo)
2001

• **STOCK SPY LA BORSA ONLINE ioProgrammo n. 84/85/86**
G. Naccarato
(Edizioni Master)

COME CREARE UN DSN

1 **Prima di ogni cosa. Dal pannello di controllo di Windows, nella cartella *"Strumenti di amministrazione"*, selezionare *"Origine dei dati (ODBC)"*.**

2 **Dalla finestra di controllo *"Amministrazione Origine dati ODBC"*, tabella *"DSN Utente"* cliccare su *aggiungi*, selezionare *"Driver Microsoft Access"* e cliccare su *Fine***

3 **Dalla Finestra *"Configurazione ODBC per Microsoft Access"* nella casella *"Nome origine dati"* digitare *STOCKDATA*, cliccare poi sul pulsante *"Seleziona"***

4 **Dalla casella *"Selezione Database"* selezionare il file *.mdb* che contiene i dati. Cliccare su *"Ok"* nella finestra in questione e in quella sottostante. Infine su *"Ok"* nella finestra principale.**

me risultato avremo:

Valore	Tempo
16.40	*2004/07/06 12:00*
16.43	*2004/07/06 17:00*

ovvero tutte le quotazioni presenti per il 6 Luglio. Il metodo *query* ha la seguente *signature:*

```
public static StockQuote[] query(String name,
            Calendar from, Calendar to, boolean intraday)
```

I parametri in input sono: *name* (nome dell'azione), *from* (data inizio), *to* (data fine) e *intraday* (*true = modalità intraday, false=modalità multiday*). Le quotazioni saranno restituite in un array di oggetti *StockQuote.* L'implementazione del metodo è la seguente:

```
String sql =
  "SELECT * FROM STOCK_QUOTE WHERE " +
  "stock = '"+name+"' AND " +
  "time >= '"+formatDateTime(from)+"' AND " +
  "time <= '"+formatDateTime(to)+"' ";
Statement st = con.createStatement();
ResultSet rs = st.executeQuery(sql);
StockQuote candidateStock = null;
ArrayList result = new ArrayList();
while (rs.next()) {
  StockQuote sq = new StockQuote();
  sq.name = rs.getString("stock");
  sq.value = rs.getDouble("value");
  sq.date = createDateTime(rs.getString("time"));
  sq.difference = rs.getDouble("difference");
  if (!intraday) {
    boolean isSameDay = candidateStock==null
      || sameDay(sq.date, candidateStock.date);
    if (isSameDay) {
      if (candidateStock==null ||
           sq.date.after(candidateStock.date))
        candidateStock = sq;
    } else {
      result.add(candidateStock);
      candidateStock = sq; }
  }
  else {
    result.add(sq);
  }
}
if (candidateStock!=null) {
  result.add(candidateStock);
}
StockQuote quotes[] =
  new StockQuote[result.size()];
for (int i = 0; i < quotes.length; i++)
{
  quotes[i] = (StockQuote) result.get(i);
}
con.commit();
return quotes;
```

Il metodo costruisce il comando SQL *SELECT* e lo inoltra al database mediante l'invocazione del metodo *executeQuery* di *Statement.* I risultati vengono successivamente letti dal *ResultSet* risultante.
All'interno del ciclo viene costruito un oggetto *StockQuote* con i dati presenti nel *ResultSet.*
A questo punto ci sarà un diverso comportamento, a secondo se la consultazione è *multiday* o *intraday.* Nel primo caso, l'oggetto *StockQuote* verrà preso in considerazione, ed alla fine aggiunto all'array *list result,* solo se è l'ultimo del giorno (ovvero se ha l'orario più alto). Nel secondo caso, tutti gli oggetti *StockQuote* estratti dal *ResultSet* saranno presi in considerazione. Infine, l'*ArrayList result,* contenente tutti gli oggetti da restituire, sarà convertito un array di *StockQuote.*

MEMORIZZIAMO LE QUOTAZIONI DEL MIB30

La classe *Mib30Dialog,* implementata lo scorso mese, può essere dotata adesso di una nuova funzionalità: memorizzare le quotazioni. Per ottenere questo comportamento aggiungeremo un bottone *"Salva"* alla dialog. Inoltre, le quotazione saranno salvate automaticamente ogni cinque minuti. In questo modo, alla fine della giornata, avremo tutte le quotazioni *intraday* relative ai titoli del Mib30.
Ecco il codice relativo al nuovo bottone *"Salva"*:

```
JButton btnSave = new JButton("Salva");
pnlBottom.add(btnSave);
btnSave.addActionListener(new ActionListener() {
  public void actionPerformed(ActionEvent e) {
    Mib30Dialog.this.saveQuote();
  }
}
);
```

Come si può notare, alla pressione del bottone *"Salva"* verrà invocato il metodo *saveQuote.* Questo, implementato all'interno della classe *Mib30Dialog,* preleverà le quotazioni aggiornate ed invocherà il metodo *save* della classe *StockDB* su ogni oggetto *StockQuote* risultante:

```
private void saveQuote() {
  try {
    connector.connect(properties);
    StockQuote quotes[] = connector.getMib30();
    stockTable.reload(quotes);
    for (int i = 0; i < quotes.length; i++) {
      StockDB.save(quotes[i]);
```

NOTA

Come importare il progetto Stock Spy in Eclipse

Inziare un nuovo progetto cliccando su *File/New Project* e poi selezionare *Java Project.* Cliccare su *Next* e digitare in *Project Name "StockQuote"* curando che il flag "*Create Separare Source and Output Folders*" sia settato. Infine cliccare su *fine.* Nella finestra dei progetti di Eclipse: cliccare con il tasto destro su *StockQuote* selezionare *Import* e poi *File System.* Utilizzando il tasto *Browse* portarsi nella directory dove abbiamo scompattato il file zip presente nel cd, selezionare i file da importare e cliccare su *finish,* curando che non sia flaggata l'opzione *Overwrite existing resource without warning* e che sia selezionato *create selected folders only.* Copiare il file *stockspy.properties* nella directory immediatamente superiore alla Src nel workspace di eclipse Curate ovviamente, di avere creato il DSN come illustrato nel box corrispondente.

```
    }
  }
  catch(Exception ex) {
    ex.printStackTrace();
  }
}
```

I dati verranno salvati comunque automaticamente ogni cinque minuti. Per far questo è necessario invocare *saveQuote* anche nel thread che si attiva periodicamente:

```
Thread t = new Thread(new Runnable() {
  public void run() {
    for(;;) {
      try {
        // Cinque Minuti
        Thread.sleep(5 * 60 * 1000);
      } catch(Exception ex) {
         ex.printStackTrace();
      }
      Mib30Dialog.this.saveQuote();
    }
  }
}
);
```

Col passare del tempo, oppure alla pressione di *"Salva"*, noterete che nella tabella *STOCK_QUOTE* del database verranno aggiunte nuove righe.

UN ESEMPIO PRATICO

Avendo a disposizione le quotazioni all'interno della tabella *STOCK_QUOTE*, sarà possibile eseguire tutte le elaborazioni che riterremo opportune. Se volete utilizzare i dati in applicazioni Java, il metodo *query* di *StockDB* risulterà molto utile. L'esempio che proporremo in questo paragrafo sarà un'applicazione che accetterà dalla riga di comando i parametri da passare al metodo *query*. Questo può essere visto come un primo passo per interrogare la nostra base di dati. L'applicazione accetterà quattro parametri: il nome dell'azione, la data della prima quotazione a cui siamo interessati, quella dell'ultima ed un flag che indica se si è interessati ad un'interrogazione *intraday (true)* o *multiday (false)*. L'implementazione è la seguente:

```
import java.util.*;
import java.io.*;
public class StockQuery {
  public static void main(String a[]) throws Exception {
    StockDB.init();
    String name = a[0];
    Calendar from =
      StockDB.createDate(a[1]);
    Calendar to =
      StockDB.createDate(a[2]);
    boolean intraday =
      a[3].equalsIgnoreCase("true")? true : false;
    to.set(to.HOUR_OF_DAY,23);
    to.set(to.MINUTE,59);
    StockQuote quotes[] =
      StockDB.query(name, from, to, intraday);
    for (int i = 0; i < quotes.length; i++)
     { System.out.println(quotes[i]); }
  }
}
```

La prima operazione è la chiamata al metodo *StockDB.init*, per ottenere la connessione con il database. Successivamente, gli argomenti passati dalla riga di comando vengono trasformati nel formato richiesto dal metodo *query*. A tale proposito, il metodo *StockDB.createDate* crea un'istanza di *Calendar* a partire da una stringa che rappresenta una data nel formato *"AAAA/MM/GG"*. Quindi, passando *"2004/07/05"* e *"2004/07/06"*, otterremo *from* e *to* che rappresenteranno rispettivamente le date *"5 Luglio 2004 ora 00:00"* e *"6 Luglio 2004 ora 23:59"*. L'orario dell'oggetto *from* è stato fissato volutamente alle 23:59 per consentire interrogazioni intraday. Ottenuti i parametri, il metodo invoca *StockDB .query* e mostra le quotazioni sulla console. Per ottenere tutte le quotazioni multiday dal 1 Gennaio 2004 al 1 Agosto 2004 dell'azione ENEL si procederà nel seguente modo:

```
> java StockQuery ENEL 2004/01/01 2004/08/01 false
```

Se invece si è interessati alle quotazioni *ENEL intraday* relative al 5 Luglio 2004 allora il comando sarà:

```
> java StockQuery ENEL 2004/07/05 2004/07/05 true
```

CONCLUSIONI

Nel corso di questi tre articoli abbiamo visto come ottenere le quotazioni delle azioni in tempo reale, ad un provider di dati, come mostrare i dati ed aggiornarli all'interno di un'applicazione e come memorizzarli su un database relazionale. Qualche idea per il futuro:

- Si può estendere la Mib30 dialog in modo che sia in grado di visualizzare tutte le azioni del Mibtel e non solo quelle del Mib30;
- Si potrebbe utilizzare un simile approccio per ottenere la quotazioni di valute, di indici o di fondi;

Abbiamo gettato le basi: a voi continuare il lavoro...

Giuseppe Naccarato

NOTA

Come compilare il progetto Stock Spy in Eclipse

Per compilare il tutto cliccare con il tasto destro sul progetto stockspy, poi selezionare *run java application*. Nella dialog Box che ne segue cliccare su *Mib30Dialog* e poi su *ok*.

Kataweb Finanza,
www.kwfinanza.kataweb.it

G. Naccarato
www.giuseppe-naccarato.com

I trucchi del mestiere

Tips & Tricks

Questa rubrica raccoglie trucchi e piccoli pezzi di codice, frutto dell'esperienza di chi programma, che solitamente non trovano posto nei manuali. Alcuni di essi sono proposti dalla redazione, altri provengono da una ricerca su Internet, altri ancora ci giungono dai lettori. Chi volesse contribuire, potrà inviare i suoi Tips&Tricks preferiti. Una volta selezionati, saranno pubblicati nella rubrica. Il codice completo dei tips è presente nel CD allegato nella directory *tips*\\ o sul Web all'indirizzo: cdrom.ioprogrammo.it.

VISUAL BASIC

COME RIPRODURRE UN CD AUDIO

All'uopo si utilizza l'API di sistema *mciSendString* che, a seconda del parametro passatogli, consente di riprodurre o arrestare la riproduzione di una traccia musicale da Cd-audio

```
Public Declare Function mciSendString Lib "winmm.dll" Alias
    "mciSendStringA" (ByVal lpstrCommand As String, ByVal
    lpstrReturnString As String, ByVal uReturnLength As Long, ByVal
    hwndCallback As Long) As Long
Sub cmdPlay_Click ()
  Dim lRet As Long, nCurrentTrack As Integer
  'Apertura del dispositivo
  lRet = mciSendString("open CD audio alias CD wait", 0&, 0, 0)
  'Imposta il formato delle Tracce (per default è millisecondi)
  lRet = mciSendString("set cd time format tmsf", 0&, 0, 0)
  'Riproduce dall'inizio del CD
  lRet = mciSendString("play cd", 0&, 0, 0)
  '...o da una traccia specifica, ad es. la 2
  'nCurrentTrack = 2
  'lRet = mciSendString("play cd from" & Str(nCurrentTrack), 0&, 0, 0)
End Sub
Sub cmdStop_Click ()
  Dim lRet As Long 'Arresta la riproduzione audio
  lRet = mciSendString("stop cd wait", 0&, 0, 0)
  DoEvents
  'Chiude il dispositivo
  lRet = mciSendString("close cd", 0&, 0, 0)
End Sub
```

CREARE NUOVE TABELLE ALL'INTERNO DI UN DB ACCESS

In questo esempio *NomeDB* è il percorso e nome del file del database *NomeTabella* il nome della tabella che si vuole creare. Per poter far funzionare il tip è necessario che all'interno del progetto sia presente un riferimento alla libreria DAO

```
Public Function CreateTable(NomeDB As String, ByVal NomeTabella
    As String) As Boolean
  On Error GoTo errorhandler
  Dim oDB As DAO.Database
  Dim td As DAO.TableDef
  Dim f As DAO.Field
  Set oDB = Workspaces(0).OpenDatabase(NomeDB)
  On Error GoTo errorhandler
  If VerificaTabella(oDB, NomeTabella) Then GoTo errorhandler
  'Crea l' oggetto tabella
  Set td = oDB.CreateTableDef(NomeTabella)
  'Aggiunge un campo ID con autoincremento
  Set f = td.CreateField("ID", dbLong)
  f.Attributes = dbAutoIncrField
  'Aggiunge il campo alla tabella
  td.Fields.Append f
  'Aggiunge la tabella al database
  oDB.TableDefs.Append td
  oDB.Close
  CreateTable = True
  Exit Function

  errorhandler:
  If Not oDB Is Nothing Then oDB.Close
  Set td = Nothing
  Set f = Nothing
End Function

Private Function VerificaTabella(oDB As Database, NomeTabella
    As String) As Boolean
  Dim td As DAO.TableDef
  On Error Resume Next
  Set td = oDB.TableDefs(NomeTabella)
  VerificaTabella = Err.Number = 0
End Function
```

JAVA

COME VISUALIZZARE UNA PAGINA HTML IN UN'APPLICAZIONE

Servono poche righe di codice e il gioco è presto fatto. Tramite il metodo *URL* si fornisce l'indirizzo della pagina web da caricare e visualizzare

```
JTextPane jTxtPane = new JTextPane();
JScrollPane jScrollPane = new JScrollPane();
jScrollPane.getViewport().add(jTxtPane);
try {
URL url = new URL("http://www.ioprogrammo.it" );
jTxtPane.setPage(url);}
catch (Exception e)
{e.printStackTrace();}
```

MOSTRARE DOCUMENTI PDF DA UN'APPLET

La soluzione proposta utilizza il metodo *showDocument()* dell'interfaccia pubblica *AppletContext* per visualizzare documenti PDF direttamente in una applet Java. È necessario che il browser adottato nel sistema sia provvisto di plug-in Acrobat Reader

```
package play.pdf;
import java.applet.*;
import java.awt.*;
import java.net.URL;
public class PDFViewer extends Applet {
Font font = new Font("Dialog", Font.BOLD, 24);
String str = "PDF Viewer";
int xPos = 5;
public String getAppletInfo() {
return "PDFViewer\n" +
"\n" +
"File creato con....\n" +
""; }
public void paint(Graphics g) {
g.setFont(font);
g.setColor(Color.black);
g.drawString(str, xPos, 50);}
public void start() {
super.start();
try {
URL url = new URL( "http://www.ioprogrammo.it/prova.pdf" );
this.getAppletContext().showDocument( url, "_blank" );
} catch (Exception e) {
System.err.println( "Errore: Impossibile visualizzare il documento!" );
}}
```

JAVA SCRIPT

COME CATTURARE I DATI DI LOGIN DA UNA FINESTRA DI POP-UP

Gli script forniti recuperano i dati immessi dall'utente nel form di una finestra di pop-up, memorizzando il loro contenuto nelle variabili *loginname* e *loginpassword*. Facendo click sul link viene passata alla funzione *go()* la pagina web che dovrà ricevere i dati di login dell'utente, quest'ultimi passati mediante la *location*. Nell'esempio che segue definiamo tre distinti file

```
File index.html
<SCRIPT LANGUAGE="javascript">
var loginname = '';
var loginpassword = '';
function windowOpen()
{ var myWindow =
  window.open('popup.htm','windowRef',width=200,height=200');
  myWindow.location.href = 'popup.htm';
  if (!myWindow.opener)
     myWindow.opener = self;}
  function go(url)
  { location.href = url + '?login=' + loginname + '&pwd='
                                                   + loginpas sword;
  }
  windowOpen();
  </SCRIPT>
  File popup.html
  <SCRIPT LANGUAGE="javascript">
  function returnDetails()
  { opener.loginname = document.myForm.loginname.value;
     opener.loginpassword = document.myForm.loginpassword.value;
     self.close();}
  </SCRIPT>
  <FORM NAME="myForm">UserID: <INPUT TYPE="password"
                                    NAME="loginname"><br>
  Password: <INPUT TYPE="password" NAME="loginpassword">
  <INPUT TYPE="button" VALUE="Invia" onClick="returnDetails()">
  </FORM>
  File pagina2.htm
  <SCRIPT LANGUAGE="javascript">
  var start = location.search.indexOf('?');
  var end = location.search.indexOf('&');
  var loginname = location.search.substring(start + 1,end);
  var loginpassword = location.search.substring(end + 1);
  alert('UserID = ' + loginname + '\nPassword = ' + loginpassword);
</SCRIPT>
```

COME ZOOMMARE UN'IMMAGINE

Lo script che segue mostra come raddoppiare le dimensioni di un'immagine operando un click sulla stessa

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
fattorea=0;
fattoreb=0;
function prova()
{  fattorea =document.images['foto'].height;
   fattoreab=document.images['foto'].width;
   document.images['foto'].height= fattorea *2;
   document.images['foto'].width= fattoreb*2; }
</SCRIPT>
</HEAD>
<BODY >
<A HREF="#"><IMG BORDER="0" NAME="foto" SRC="images/003.gif"
WIDTH="150" HEIGHT="121" onClick="prova()"></A>
</BODY>
</HTML>
```

C#

COME CANCELLARE UNA DIRECTORY IN MODO RICORSIVO

Si tratta di una semplice operazione che è possibile realizzare invocando il metodo *Delete* della classe *Directory* e impostando il secondo argomento di tale metodo a *true*

```
using System.IO;
class TestDirectory
{ public static void Main()
{ Directory.Delete("C:\\testc_sharp", true);}}
```

LA CANCELLAZIONE DI UN FILE DALL'HARD-DISK

Cancellare un file è molto semplice: basta utilizzare il metodo *Delete* della classe *File*, come mostrato nell'esempio che segue

```
using System.IO;
class TestFile
{ public static void Main()
{ File.Delete( "c:\\dog.jpg" );}}
```

COME LEGGERE IL CONTENUTO DI UN GENERICO FILE DI TESTO

La funzione *LeggiFile* riceve come parametro il percorso del file di testo da leggere e restituisce una stringa contenente l'intero contenuto del file. In aggiunta implementa la gestione degli errori e verifica l'esistenza del file

```
using System;
using System.IO;
public string LeggiFile(string sPath)
{ string sContent = "";
   try {
     if (File.Exists(sPath)) {
        // apre il file in lettura
        StreamReader srTesto = File.OpenText(sPath);
        sContent = srTesto.ReadToEnd();
        srTesto.Close(); }
     } catch (Exception exRet) {
        sContent = "ERRORE: " + exRet.Message; }
     return(sContent);
}
```

DELPHI

REALIZZARE UN SEMPLICE BLOCCO NOTE

Per realizzare un semplicissimo Blocco Note con Delphi è sufficiente posizionare su di una form un componente *Tmemo* che chiameremo *Memo1*. Per salvare il testo su un file *.TXT* è sufficiente scrivere il seguente codice:

```
Memo1.Lines.SaveToFile('APPUNTI.TXT');
```

Per aprire il file di testo e riaverlo nel componente *Tmemo* scrivere invece il seguente codice:

```
Memo1.Lines.LoadFromFile('APPUNTI.TXT');
```

SPOSTARE E RIDIMENSIONARE UN COMPONENTE A RUNTIME

Il pezzo di codice, unito all'evento *MouseMove* di un generico *TButton "Button1"*, permette di spostarlo e ridimensionarlo. Per spostarlo bisogna trascinare il mouse col tasto sinistro cliccato, mentre per ridimensionarlo bisogna cliccare e spostare tenendo premuto *Ctrl*

```
{ ,-------------------------------, }
{|  ObjPas-DragAndDrop by X-3mE'89  |}
{|  distribuito secondo i  termini  |}
{| della GNU General Public License |}
{|,-------------------------------,|}
{|  http://exxtreme.altervista.org  |}
{ '-------------------------------' }
procedure TForm1.Button1MouseMove(Sender: TObject; Shift:
                                    TShiftState; X, Y: Integer);
var delta:longint; { spostamento }
begin
 if ssLeft in Shift then begin { se il bottone sinistro e' cliccato... }
   if(ssCtrl in Shift) then begin { e anche Ctrl... }
      { ... allora ridimensiona }
      delta:=Button1.Left+x-Button1.Width;
      { la linea qui sotto serve per dare l'effetto }
      { di "ancoraggio" come in Delphi o Lazarus al }
      { nostro bottone }
      while(delta mod 16 <> 0) do delta:=delta+1;
      Button1.Width:=Button1.Width+delta;
      delta:=Button1.Top+y-Button1.Height;
      while(delta mod 16 <> 0) do delta:=delta+1;
      Button1.Height:=Button1.Height+delta;
   end
   else begin
      { ... altrimenti sposta }
      delta:=Button1.Left+X-(Button1.Width div 2);
      while(delta mod 16 <> 0) do delta:=delta+1;
      Button1.Left:=delta;
      delta:=Button1.Top+Y-(Button1.Height div 2);
      while(delta mod 16 <> 0) do delta:=delta+1;
      Button1.Top:=delta;
   end;
 end;
end;
```

CONVERSIONE DELLE CIFRE IN LETTERE

La funzione *CifraLettere*, così come lo stesso nome suggerisce, consente di trasformare una cifra (fino al migliaio) in lettere.
Si tratta di un trucchetto molto utile nel caso di applicazioni di magazzino, fatture, ecc.

```
function CifraLettere(cifra: string): string;
```

IL TIP DEL MESE

PORTING DI UN FILE XML IN UN DB

Nel Cd-Rom allegato alla rivista (e sul sito web *www.ioprogrammo.it*) è presente l'esempio completo che consente di migrare un piccolo file XML in un database di prova. In aggiunta è presente anche un file TXT in cui sono riportare le due funzioni principali. L'applicazione preleva i nomi dei nodi principali del file XML (che hanno lo stesso nome delle tabelle del database) e inserisce i record all'interno delle tabelle del database attraverso due cicli innestati.

Tip fornito dal sig. N.Barbieri

```
procedure TForm1.updateDatabase(Sender: TObject);
var
 root : IXMLNode;
 nodo1,nodo2 :IXMLNode;
 i1,i2,numNodi,numRECORD : Integer;
 ret_val,flag:Integer;
begin
 i1:=0;
 i2:=0;
 flag:=0;
 XMLDocument1.Active:=True;
 root:=XMlDocument1.DocumentElement;
 numNodi:=root.ChildNodes.Count;
 nodo1:=root.ChildNodes.First;
 if numNodi>0 then
   begin
     repeat
       numRECORD:=nodo1.ChildNodes.Count;
       ADOTable1.TableName:=nodo1.NodeName;
       if numRECORD>0 then
         begin
         nodo2:=nodo1.ChildNodes.First;
           repeat
             ret_val:=InsertDatabase(nodo2);
                     //chiama la funzione di inserimento del nodo
             if ret_val=-1 then
               begin
                     flag:=1;
                 ShowMessage('Si è verificata un errore di inserimento');
               end;
             nodo2:=nodo2.NextSibling;
             i2:=i2+1;
           until i2>=numRECORD;
         end;
           i2:=0;
           nodo1:=nodo1.NextSibling;
           i1:=i1+1;
     until i1>=numNodi;
   end;
   if flag=0 then
     begin
       ShowMessage('Update eseguito con successo');
     end;
end;
function TForm1.InsertDataBase(nodo2:IXMLNode):Integer;
var
I:Integer;
i3:Integer;
nodo3:IXMLNode;
flag:Integer;
begin
 i3:=0;
 flag:=0;
 ADOTable1.Open;
 with ADOTable1 do
 begin
   Insert;
   for I:=0 to nodo2.ChildNodes.Count -1 do
     begin
       if (i3=0) then
             begin
               nodo3:=nodo2.ChildNodes.First;
               FieldByName(nodo3.NodeName).Value:=
                                             nodo3.NodeValue;
             end
           else
             begin
               nodo3:=nodo3.NextSibling;
               FieldByName(nodo3.NodeName).Value:=
                                             nodo3.NodeValue;
             end;
       i3:=i3+1;
     end;
     try
       ADOTable1.Post;
     except
       on E:Exception do
       flag:=-1
     end;
 end;
 ADOTable1.Close;
 result:=flag;
end;
```

```
var lettere: string; valore: integer;
begin
valore := 0;
//Trasformazione MIGLIAIA
if Length(cifra) = 4 then
begin
 valore := StrToInt(AnsiLeftStr(cifra, 1));
 case valore of
  1: lettere := 'mille';
  2: lettere := 'duemila';
  3: lettere := 'tremila';
  4: lettere := 'quattromila';
  5: lettere := 'cinquemila';
  6: lettere := 'seimila';
  7: lettere := 'settemila';
  8: lettere := 'ottomila';
  9: lettere := 'novemila';
 end;
end;
//Trasformazione CENTINAIA
if (Length(cifra) = 4) or (Length(cifra) = 3) then
begin
```

```
 if Length(cifra) = 4 then valore := StrToInt(AnsiRightStr(
                                                    AnsiLeftStr(cifra, 2),1));
 if Length(cifra) = 3 then valore := StrToInt(AnsiLeftStr(cifra, 1));
 case valore of
  1: lettere := lettere +'cento';
  2: lettere := lettere +'duecento';
  3: lettere := lettere +'trecento';
  4: lettere := lettere +'quattrocento';
  5: lettere := lettere +'cinquecento';
  6: lettere := lettere +'seicento';
  7: lettere := lettere +'settecento';
  8: lettere := lettere +'ottocento';
  9: lettere := lettere +'novecento';
 end;
end;
//Trasformazione DECINE
if (Length(cifra) >= 2) and (StrToInt(AnsiLeftStr(AnsiRightStr(cifra,
                                                              2), 1)) <> 1)
 and (StrToInt(AnsiRightStr(cifra, 1)) <> 1) and (StrToInt(
                                         AnsiRightStr(cifra, 1)) <> 8) then
begin
 valore := StrToInt(AnsiLeftStr(AnsiRightStr(cifra, 2), 1));
 case valore of
  2: lettere := lettere +'venti';
  3: lettere := lettere +'trenta';
  4: lettere := lettere +'quaranta';
  5: lettere := lettere +'cinquanta';
  6: lettere := lettere +'sessanta';
  7: lettere := lettere +'settanta';
  8: lettere := lettere +'ottanta';
  9: lettere := lettere +'novanta';
 end;
end else
if (Length(cifra) >= 2) and (StrToInt(AnsiRightStr(cifra, 1)) = 1)
                          or (StrToInt(AnsiRightStr(cifra, 1)) = 8) then
begin
 valore := StrToInt(AnsiLeftStr(AnsiRightStr(cifra, 2), 1));
 case valore of
  2: lettere := lettere +'vent';
  3: lettere := lettere +'trent';
  4: lettere := lettere +'quarant';
  5: lettere := lettere +'cinquant';
  6: lettere := lettere +'sessant';
  7: lettere := lettere +'settant';
  8: lettere := lettere +'ottant';
  9: lettere := lettere +'novant';
 end;
end else
if (Length(cifra) >= 2) and (StrToInt(AnsiLeftStr(AnsiRightStr(
                                                    cifra, 2), 1)) = 1) then
begin
 valore := StrToInt(AnsiRightStr(cifra, 2));
 case valore of
  10: lettere := lettere +'dieci';
  11: lettere := lettere +'undici';
  12: lettere := lettere +'dodici';
  13: lettere := lettere +'tredici';
  14: lettere := lettere +'quattordici';
  15: lettere := lettere +'quindici';
  16: lettere := lettere +'sedici';
  17: lettere := lettere +'diciassette';
  18: lettere := lettere +'diciotto';
  19: lettere := lettere +'diciannove';
 end;
end;
//Trasformazione UNITA'
if (Length(Cifra) >= 1) then
 if ((Length(cifra) >= 2) and (StrToInt(AnsiLeftStr(AnsiRightStr(
                                            cifra, 2), 1)) > 1)) then
 begin
  valore := StrToInt(AnsiRightStr(cifra,1));
  case valore of
   1: lettere := lettere +'uno';
   2: lettere := lettere +'due';
   3: lettere := lettere +'tre';
   4: lettere := lettere +'quattro';
   5: lettere := lettere +'cinque';
   6: lettere := lettere +'sei';
   7: lettere := lettere +'sette';
   8: lettere := lettere +'otto';
   9: lettere := lettere +'nove';
  end;
 end;
 Result := lettere;
end;
```

Un progetto elettronico per costruire un sistema wireless

Un Radio Modem per PC

Ci avventureremo nel mondo della trasmissione radio digitale a lunga distanza: scopriremo come realizzare un Modem a radiofrequenza capace di una portata di molte centinaia di metri

Utilizza questo spazio per le tue annotazioni

In questo articolo, dopo una brevissima trattazione teorica, proporremo un semplice sistema per la trasmissione di dati digitali a distanza. Utilizzeremo semplici moduli a radiofrequenza nella banda dei 433 MHz. Tutte le informazioni necessarie alla realizzazione del sistema saranno contenute in questo articolo, partendo dal circuito elettronico, per giungere al software di controllo.

LA MASSIMA VELOCITÀ DI TRASMISSIONE

Il teorema di Shannon-Hartley definisce la massima quantità di dati digitali esenti da errori che può essere trasmessa attraverso un canale di comunicazione, dotato di una specifica larghezza di banda. Individuando con C la capacità di trasmissione teorica in bit per secondo, con *W* la banda passante in Hertz e con S/N *(Signal/Noise)* il rapporto tra segnale e rumore, Shannon ed Hartley scrivono:

$$C \leq W \log_2 (1+S/N)$$

Applicando questa formula ai parametri di una linea telefonica immaginaria, generica e puramente teorica, considerando un rapporto S/N di 25 dbm ed una banda passante di 3,5 KHz, calcolando il limite di Shannon abbiamo:

Fig. 1: Visione d'insieme del ricevitore

C=29 Kbit/Sec

Corrispondente alla massima quantità d'informazioni che possono essere trasmesse dalla linea in questione senza errori. Conoscendo i parametri di un qualsiasi sistema di trasmissione, possiamo calcolare il relativo limite di Shannon.

TRASMISSIONE DATI VIA RADIO

Esistono innumerevoli metodi di trasmissione dati via radio, e altrettanti possibili protocolli di comunicazione per ottenere lo scambio d'informazioni tra un trasmettitore ed un ricevitore. Un metodo potrebbe essere quello di utilizzare le normali linee di trasmissione e ricezione della porta seriale, al fine di ricavarne i segnali necessari a pilotare un circuito a radiofrequenza. Utilizzando questo metodo avremmo l'indubbio vantaggio di potere utilizzare la UART *(Universal Asynchronous Receiver Transmitter)* contenuta all'interno della porta seriale, ma avremmo anche la necessità di farci carico di un circuito aggiuntivo d'interfaccia con il modulo trasmettitore.
In questa sede vogliamo proporre la realizzazione di un sistema di trasmissione dati a radiofrequenza dotato di protocollo di trasmissione 'proprietario'.
Il trasmettitore del nostro sistema è basato sul modulo *TX-SAW-433/s-Z*, della Aurel (*www.aurel.it*) ed è destinato ad un impiego dove sia necessaria la modulazione *ON-OFF* di una portante per la trasmissione di segnali digitali ed è conforme alle direttive europee sull'argomento. Lo schema elettrico viene riportato in queste pagine ed è già stato commentato nell'articolo della sezione elettronica di ioProgrammo del numero precedente. Il ricevitore è stato realizzato utilizzando il modulo *4M50RR30SF*, sempre della Aurel, assemblando i pochi componenti

elettronici per mezzo della apparecchiatura PCExplorer light (*www.pcexplorer.it*).

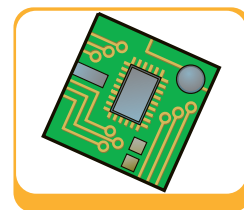
IL PROTOCOLLO DI TRASMISSIONE

Il nostro protocollo di trasmissione è basato su un treno di impulsi che sono successivamente inviati al modulo trasmettitore a radiofrequenza. La modulazione del segnale avviene attraverso la tecnica PWM *(Pulse Width Modulation)*, per mezzo della quale viene modificata la lunghezza dell'impulso a seconda che si desideri trasmettere un dato binario '1' oppure '0'. In particolare, la lunghezza di un impulso corrispondente ad un valore '0' è doppia rispetto alla lunghezza di un impulso corrispondente ad un '1'. Per semplicità, trasmetteremo singoli byte di otto bit. Ogni byte è preceduto da un bit di *START*, corrispondente ad un impulso lungo (ovvero uno '0'), cui segue otto bit di dati con il metodo di modulazione appena descritto, ed infine una sequenza d'impulsi brevi ('1') che determinano la condizione di *STOP*. Il numero di questi impulsi è fissato in modo tale da ottenere un intervallo tra la trasmissione di due blocchi di dati. Il motivo di ciò risiede nel fatto che il ricevitore può ipoteticamente mettersi in ricezione in qualunque momento della sequenza di trasmissione. Quindi, per stabilire quando uscire dalla condizione di *STOP*, deve poter rilevare una sequenza d'impulsi la cui lunghezza sia superiore al numero di bit di dati più il numero di bit di *START*. Nel nostro caso, il numero di bit di *STOP* dovrà essere superiore a 8+1=9 bit. Avrete notato che la frequenza di modulazione è variabile e che la sequenza di bit di *STOP* è stata definita in impulsi '1', ovvero lunghi la metà di uno '0' per diminuire il tempo di attesa tra un byte e l'altro.

REALIZZAZIONE DEL SISTEMA

Il trasmettitore ed il ricevitore sono elettronicamente identici a quelli utilizzati per il radiocomando ad un canale proposto nel numero precedente di ioProgrammo. Per facilitarne la realizzazione è possibile assemblare i componenti senza saldature per mezzo della apparecchiatura PC Explorer light. Il lettore potrà reperire i pochi componenti elettronici

Fig. 6: ***Il protocollo di trasmissione prevede un bit di START, otto bit di dati, e una sequenza di impulsi di STOP***

che utilizzeremo per la costruzione del sistema, in qualunque negozio di componenti elettronici, oppure per corrispondenza presso la Elisys. Sul lato sinistro dello schema si possono notare le connessioni alle linee relative alla porta seriale e di alimentazione di PC Explorer light.

IL SOFTWARE DI CONTROLLO

Il software di controllo del trasmettitore permette di inviare al modulo ricevitore tutti i segnali necessari per una trasmissione dati a radiofrequenza. Come di consueto, analizzeremo soltanto le parti salienti del software di controllo. Per ogni eventuale chiarimento, delucidazione o consiglio, l'autore è lieto di rispondere a qualunque domanda all'indirizzo *luca.spuntoni@ioprogrammo.it*. Il programma è contenuto in un'unica unit: gli elementi esterni al programma sono riportati nella clausola *Uses*, come riportato di seguito.

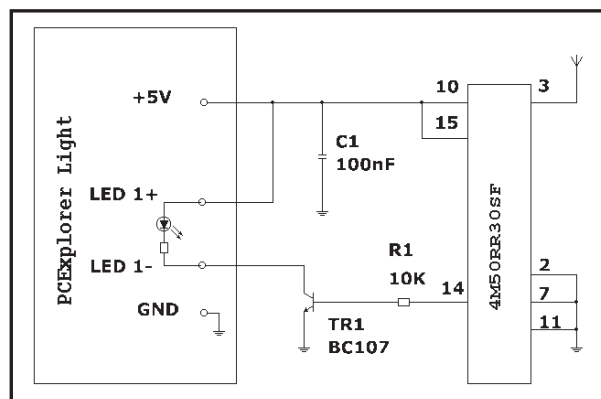



Fig. 4: ***Lo schema elettrico del ricevitore è molto semplice, stabile ed affidabile***

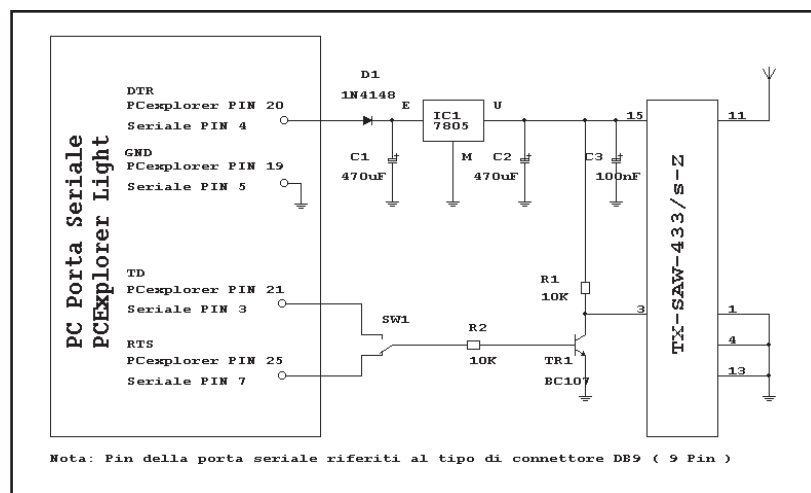



Fig. 5: ***Il trasmettitore ha una portata effettiva di oltre cinquecento metri senza ostacoli ed oltre settanta tra gli edifici***

```
unit SpuntoRadioModemUnit;
interface
uses
  Windows, Messages, SysUtils, Variants, Classes,
  Graphics, Controls, Forms, Dialogs, ExtCtrls,
  SpuntoLedComponent, jpeg, Buttons, StdCtrls,
  SpuntoHyperLabel, Spin;
```

Viene definito un tipo particolare di array, formato da otto elementi *Boolean*, che ha lo scopo di rendere possibile la conversione da un byte binario in una sequenza di valori logici da inviare alla trasmissione a radiofrequenza, oltre che per ulteriori funzioni di conversione interne.

```
type
//************ Port related arrays ************//
  TPortArray= Array[0..7] of Boolean; // Array of Port bits
// ************* MAIN CLASS *************//
```

La classe principale è riportata parzialmente di seguito, per evidenziare le procedure contenute all'inteno della classe stessa.

```
 TSpuntoRadioModemForm = class(TForm)
   procedure POWERONbuttonClick(Sender: TObject);
   procedure FormCreate(Sender: TObject);
   procedure WritePort(PortAddress, PortData:word);
   function ReadPort(PortAddress: word): word;
   procedure RadioButtonCOM1Click(Sender: TObject);
   procedure RadioButtonCOM2Click(Sender: TObject);
   procedure ReadAllPorts;
   …
   procedure TXDataSpinEditChange(Sender: TObject);
 private
   { Private declarations }
 public
   { Public declarations }
   CombaseAddress:Word;
   MCRAddress,LCRAddress,MSRAddress,
                                   TransmissionWord:Word;
   MCR,LCR,MSR,TransmissionArray:TPortArray;
   EnablePortWriting: Boolean;
   RTS,CTS,DSR,CD,DTR,RI:Boolean;
   TransmissionbitCounter:Integer
  end;
```

La procedura *POWERON* permette l'accensione del trasmettitore, attivando la linea DTR *(Data Terminal Ready)* deputata a garantire l'alimentazione elettrica del circuito.

```
procedure TSpuntoRadioModemForm.
                    POWERONbuttonClick(Sender: TObject);
VAR
  MCRWord:Word;
begin
  SquareWaveTimer.Enabled:=(POWERONbutton.Down
                    and TXSquareWaveSpeedButton.Down);
  Timer1.Enabled:=POWERONbutton.Down;
  ReadAllPorts;
  If DTR then DTRSpeedButton.Down:=True else
                            DTRSpeedButton.Down:=False;
  If RTS then RTSSpeedButton.Down:=True else
                            RTSSpeedButton.Down:=False;
MCR[0]:=POWERONbutton.Down;
if TXSquareWaveSpeedButton.down=False then
                                        MCR[1]:=False;
// RTS Line Codifico MCR in Word e lo invio alla porta
EncodePortArray(MCRWord,MCR);
WritePort(MCRAddress,MCRWord);
end;
```

Appena viene lanciato il programma e creata la *form*, viene eseguita la procedura *FormCreate* che provvede ad impostare i parametri della porta seriale in uso (*COM1* nel nostro caso) impostare i timer relativi all'aggiornamento degli elementi grafici della *Form (Timer1)*, alla modulazione del segnale *(ModulationTimer)* ed alla generazione della forma d'onda *(SquareWaveTimer)*, oltre che ad impostare tutte le variabili globali del programma.

```
procedure TSpuntoRadioModemForm.FormCreate(
                                     Sender: TObject);
begin
// Port Setup ( COM 1 )
  EnablePortWriting:= True;
  CombaseAddress:=$3f8;
  MCRAddress:=CombaseAddress+4;
  LCRAddress:=CombaseAddress+3;
  MSRAddress:=CombaseAddress+6;
// Timer Setup
  Timer1.Enabled:=False;
  Timer1.Interval:=10;
 // Buttons setup
 ReadAllPorts;
 If DTR then DTRSpeedButton.Down:=True else
                           DTRSpeedButton.Down:=False;
 If RTS then RTSSpeedButton.Down:=True else
                           DTRSpeedButton.Down:=False;
 // Waveform and modulation Timer Interval
 SquareWaveTimer.Interval:=SquareWaveScrollBar.
                                            Position;
 ModulationTimer.Interval:=SquareWaveScrollBar.
                                      Position div 2;
 //Transmission Enable at startup or not?
 SquareWaveTimer.Enabled:=
                       TXSquareWaveSpeedButton.Down;
 //Encoding transmission word
 TransmissionWord:=TXDataSpinEdit.Value;
 ExtractPortArray(TransmissionWord, TransmissionArray);
 TransmissionbitCounter:=-1; //Start bit
end;
```

La scrittura della porta seriale a basso livello avviene per mezzo della procedura *WritePort*, che comprende una breve sequenza in *Assembler* che semplicemente scrive il dato *PortData* nell'indirizzo *PortAddress*.

```
procedure TSpuntoRadioModemForm.WritePort(
                             PortAddress,PortData:word);
// Write PortData over PortAddress if Port Writing is enabled
```

SUL WEB

L'apparecchiatura PC Explorer light è prodotta e commercializzata dalla Elisys s.r.l. e può essere acquistata al prezzo di € 213,60 nella versione light, € 99 nella versione basic e € 69 in kit (IVA inclusa) sul web (www.pcexplorer.it), inviando una e-mail all'indirizzo pcexplorer@elisys.it, oppure telefonicamente al numero 0823/468565 o via Fax al: 0823/495483.

NOTA

PRECAUZIONI

Prima di collegare il circuito al nostro PC occorre verificare la nostra realizzazione con attenzione per assicurarci che tutto sia stato collegato come previsto.

METODI PER AUMENTARE LE PRESTAZIONI DI UNA LINEA

Per completezza d'informazione possiamo aggiungere che esistono metodi per aumentare le prestazioni di una linea, principalmente incrementando il rapporto segnale/rumore (S/N), ma anche utilizzando sistemi d'analisi del segnale al prezzo però di una maggiore potenza di calcolo.

```
begin
  If EnablePortWriting then begin
     PortData := (PortData*256)+PortData;
     asm
     Mov ax,PortData
     Mov dx,PortAddress
     Out dx,ax
     end;
  End;
end;
```

La funzione che segue invece provvede a leggere la porta dall'indirizzo fisico *PortAddress*.

```
function TSpuntoRadioModemForm.ReadPort(
                              PortAddress: word): word;
var
  ReadPortData: word;
begin
     asm
     Mov dx,PortAddress
     In ax,dx
     Mov ReadPortData,ax
     end;
Result := Byte(ReadPortData);
end;
```

Nel caso il lettore voglia utilizzare una porta diversa da *COM1*, è possibile variarne l'indirizzo fisico nella procedura che segue.

```
procedure TSpuntoRadioModemForm.
             RadioButtonCOM1Click(Sender: TObject);
begin
// Port Setup ( COM 1 )
  EnablePortWriting:= True;
  CombaseAddress:=$3f8;
  MCRAddress:=CombaseAddress+4;
  LCRAddress:=CombaseAddress+3;
  MSRAddress:=CombaseAddress+6;
end;
```

La lettura delle tre porte fisiche che compongono la seriale avviene per mezzo della procedura *ReadAllPorts*: in questa fase vengono anche impostate le variabili globali del sistema, relative allo stato delle linee asincrone del dispositivo.

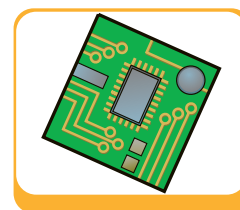
```
procedure TSpuntoRadioModemForm.ReadAllPorts;
//Readr all COM Ports related to selected serial port
Var
  MCRWord,LCRWord,MSRWord:Word;
Begin
  MCRWord:=Readport(MCRAddress);
  LCRWord:=Readport(LCRAddress);
  MSRWord:=Readport(MSRAddress);
  ExtractPortArray(MCRWord,MCR);
  ExtractPortArray(MSRWord,MSR);
  ExtractPortArray(LCRWord,LCR);
  RTS:=MCR[1];       // OUT
  CTS:=MSR[4];       // IN
  DSR:=MSR[5];       // IN
  CD :=MSR[7];       // IN
  DTR:=MCR[0];       // OUT
  RI :=MSR[6];       // IN
End;
```

Timer1Timer ha il compito di tenere aggiornato lo stato logico dei componenti visuali della *form*.

```
procedure TSpuntoRadioModemForm.Timer1Timer(
                                   Sender: TObject);
begin
ReadAllPorts;
If RTS then LedRTS.LedOn else LedRTS.LedOff;
If CTS then LedCTS.LedOn else LedCTS.LedOff;
If DSR then LedDSR.LedOn else LedDSR.LedOff;
If CD  then LedCD.LedOn else  LedCD.LedOff;
If DTR then LedDTR.LedOn else LedDTR.LedOff;
If RI  then LedRI.LedOn else  LedRI.LedOff;
//RF OUT LED
If RTS then RFOUTSpuntoLed.LedOn else
                              RFOUTSpuntoLed.LedOff;
end;
```

È possibile variare manualmente lo stato delle linee asincrone DTR ed RTS, attraverso le due procedure che seguono.

```
procedure TSpuntoRadioModemForm.DTRSpeedButtonClick(
                                   Sender: TObject);
VAR
MCRWord:Word;
begin
ReadAllPorts;
MCR[0]:=DTRSpeedButton.Down;
//Codifico MCR in Word e poi lo invio alla porta
EncodePortArray(MCRWord,MCR);
WritePort(MCRAddress,MCRWord);
end;
procedure TSpuntoRadioModemForm.RTSSpeedButtonClick(
                                   Sender: TObject);
VAR
MCRWord:Word;
begin
ReadAllPorts;
MCR[1]:=RTSSpeedButton.Down;
//Codifico MCR in Word e poi lo invio alla porta
EncodePortArray(MCRWord,MCR);
WritePort(MCRAddress,MCRWord);
end;
```

Il dato che desideriamo trasmettere viene memorizzato all'interno di *TXDataSpinEdit*, qualora questo valore cambi, l'event handler seguente provvede a convertire il nuovo parametro ed a memorizzarlo in

ACQUISTARE PC EXPLORER LIGHT

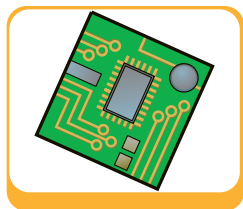
L'apparecchiatura PC Explorer light è prodotta e commercializzata dalla Elisys s.r.l. e può essere acquistata al prezzo di € 213,60 nella versione light, € 99 nella versione basic e € 69 in kit (IVA inclusa) sul web all'indirizzo www.pcexplorer.it oppure inviando una e-mail all'indirizzo pcexplorer@elisys.it, od anche telefonicamente al numero 0823/468565 o via Fax al: 0823/495483.

TRASMETTITORE

N1	Modulo TX-SAW-433/s-Z
N1	7805
N1	Transistor BC 107
N1	Diodo 1N4148
N2	Resistenza 10Kohm 1/4W
N2	Condensatori 470 uF 15V
N1	Condensatore 100 nF
N1	Deviatore a levetta

RICEVITORE

N1	PCExplorer light
N1	Modulo 4M50RR30SF
N1	Transistor BC 107
N1	Resistenza 10Kohm 1/4W
N1	Condensatore 100 nF

Per la realizzazione è possibile utilizzare PCExplorer light, l'apposito kit di montaggio, oppure una piastra millefori per montaggi sperimentali.

TransmissionArray per la successiva trasmissione.

```
procedure
TSpuntoRadioModemForm.TXDataSpinEditChange(
                                        Sender: TObject);
begin
 TransmissionWord:=TXDataSpinEdit.Value;
 ExtractPortArray(TransmissionWord, TransmissionArray);
end;
```

Il cuore del programma risiede nel codice riportato di seguito, nel quale viene generata la sequenza di segnali che sono successivamente inviati al trasmettitore. La procedura è l'event handler dell'evento generato da *SquareWaveTimer.* Il byte da trasmettere è memorizzato all'interno dell'array *TransmissionArray* sotto forma di una sequenza d'elementi Boolean. In pratica avviene il conteggio di quanti bit sono stati trasmessi attraverso l'utilizzo della variabile globale *TransmissionbitCounter:* inizialmente viene generato il bit di *START,* poi vengono trasmessi gli otto bit di dati, dal bit meno significativo ed infine viene generata la sequenza di bit di *STOP.*

Fig. 8: L'immagine mostra il trasmettitore completamente assemblato, privo di contenitore

```
procedure TSpuntoRadioModemForm.
              SquareWaveTimerTimer(Sender: TObject);
VAR
  MCRWord:Word;
  TransmissionBit:boolean;
begin
  ReadAllPorts;
  MCR[1]:=Not(RTS);
  if TXSquareWaveSpeedButton.down=False then
                                        MCR[1]:=False;
  //Codifico MCR in Word e poi lo invio alla porta
  EncodePortArray(MCRWord,MCR);
  WritePort(MCRAddress,MCRWord);
  if (TransmissionbitCounter=-1) then
  begin
    TransmissionBit:=False;     //Start bit
  end
  else begin
    TransmissionBit:=True;      //One of the Stop bits
  end;
  if ((TransmissionbitCounter>=0)and(
                    TransmissionbitCounter<=7)) then
     TransmissionBit:=TransmissionArray[
                 TransmissionbitCounter];  //Data bit
  if (TransmissionBit and (Not RTS)) then
                         ModulationTimer.Enabled:=True;
  TXbitLabel.Caption:=inttostr(TransmissionBitCounter);
  TXBitValueLabel.Caption:=booltostr(TransmissionBit);
  if (Not RTS) then inc(TransmissionBitCounter);
  if TransmissionBitCounter=20 then
                              TransmissionBitCounter:=-1;
end;
procedure TSpuntoRadioModemForm.
             ModulationTimerTimer(Sender: TObject);
VAR
  MCRWord:Word;
begin
 ReadAllPorts;
 MCR[1]:=False;          // RTS to LOW
 //Codifico MCR in Word e poi lo invio alla porta
 EncodePortArray(MCRWord,MCR);
 WritePort(MCRAddress,MCRWord);
 ModulationTimer.Enabled:=False;
end;
```

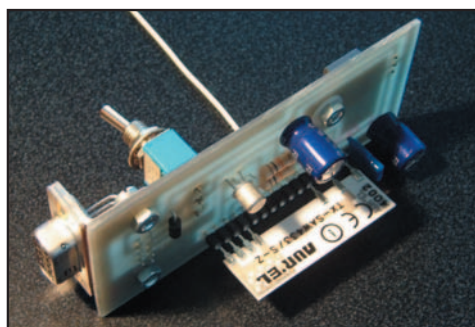
INSTALLAZIONE ED ESECUZIONE

L'installazione avviene automaticamente eseguendo il file: *SpuntoRadioModem_Install.exe.* Il file eseguibile d'installazione provvede automaticamente a predisporre la creazione di un'unica cartella configurabile dall'utente. Al fine di consentire l'esecuzione del software, poiché questo accede direttamente all'hardware della macchina, viene installato il driver: *'PortTalk - A Windows NT/2000/XPI/O Port Device Driver Version 2.2'* che può essere anche scaricato dal sito *www.beyondlogic.org/ porttalk/porttalk.htm.* Il file *'Porttalk22.zip'* contiene tutte le istruzioni necessarie all'utilizzo corretto del driver, nonché una notevole mole di informazioni relative all'accesso delle porte hardware del PC: viene fornito inoltre il codice sorgente completo delle applicazioni. Per quanto riguarda l'utilizzo del programma proposto in questa sede sotto Win 2000/NT/XP vengono estratti i file contenenti il driver *'Porttalk'* nella directory del programma da utilizzare e viene copiato il file *'porttalk.sys'* nella directory di sistema.

CONCLUSIONI

Un doveroso ringraziamento è dovuto alla AUREL S.p.A per la cortese autorizzazione alla pubblicazione della documentazione relativa ai moduli *TX-SAW 433/s-Z434* e *RX-4M50RR30SF.* Nonostante quanto esposto in queste pagine sia stato debitamente verificato e collaudato, tuttavia viene riportato a scopo illustrativo.

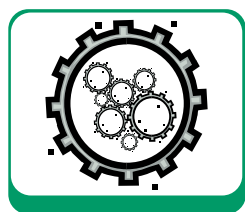
L'editore e l'autore non sono da considerare responsabili per eventuali conseguenze derivanti dall'utilizzo di quanto esposto soprattutto per la tipologia e la complessità dell'argomento.

Luca Spuntoni

BIBLIOGRAFIA

- **A MATHEMATICAL THEORY OF COMMUNICATION (The Bell System Technical Journal) 1948**
- **TX-SAW 433/s-Z 434 MHz OOK TRANSMITTER (Aurel) 2001**
- **HIGH PERFORMANCE & LOW COST RECEIVER RX-4M50RR30SF Aurel 2001**
- **PRACTICAL RF CIRCUIT DESIGN FOR MODERN WIRELESS SYSTEM, VOL I e II *Besser-Gilmore* (Artech House Books) 2003**
- **RF DESIGN GUIDE *P. Vizmuller* (Artech House Books) 1995**

Utilizzare Java Advanced Imaging per manipolare le immagini

Fotoritocco in Java

Avete mai desiderato creare il vostro Photoshop in Java? Con Java Advanced Imaging è possibile: grazie a questa potente libreria di Sun è possibile eseguire le più avanzate manipolazioni di immagini

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
Conoscenze base di Java

Software
Java2 SDK 1.4.2

Impegno

Tempo di realizzazione

Nel corso di questo e dei prossimi articoli affronteremo lo sviluppo di un'applicazione per la manipolazione delle immagini dotata delle principali funzioni: inversione dei colori, sfocatura, individuazione dei contorni e tanto altro. Tutte queste funzionalità saranno fornite all'applicazione dalle API *JAI* (*Java Advanced Imaging*) di Sun.

COSTRUIRE L'INTERFACCIA UTENTE

I manuali di programmazione sconsigliano di partire dallo sviluppo dell'interfaccia utente, in quanto la prima fase dovrebbe essere quella della pianificazione e progettazione dell'applicazione; è un buon consiglio, che in generale si dovrebbe seguire. Nello sviluppo di *JAIPhoto* si sovvertirà questo consiglio, e partiremo proprio dall'interfaccia utente, ma per alcuni buoni motivi:

- la progettazione è già stata fatta a priori, e verrà mostrata man mano che si realizzano le cose, invece che tutta a priori;
- con l'utilizzo di strumenti come Eclipse (*www.eclipse.org*) è facile realizzare parti di codice di prova, per poi evolverle con il tempo utilizzando il *refactoring*.

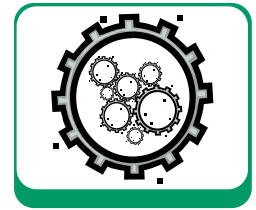
La classe principale dell'applicazione è *it.bigatti.jaiphoto.ui.Main*, che implementa un semplice metodo *main()* per l'avvio del programma:

```
public static void main(String[] args) {
  new Main(); }
```

Il costruttore, in seguito, chiama il metodo *createUI()*, che si occupa di creare l'interfaccia utente. Questa è formata da una finestra di tipo *JFrame* dal titolo *"JAI Photo"* e contiene un pannello principale configurato per utilizzare i layout manager *BorderLayout*. Come noto, questo layout manager dispone quattro componenti (*nord*, *sud*, *est* e *ovest*) concedendo loro lo spazio desiderato in orizzontale (per i componenti *nord* e *sud*) e in verticale (per i componenti *est* e *ovest*). Tutto lo spazio centrale viene invece dedicato al componente di *centro*. Nel nostro caso, quest'ultimo è un oggetto *it.bigatti.jaiphoto.ui.ZoomableImagePanel*, oggetto che conterrà l'immagine in fase di modifica. A sud è invece presente l'oggetto *infoBar*, che contiene alcuni elementi informativi, come la barra di avanzamento dell'operazione in corso, lo zoom corrente e la dimensione dell'immagine. La *JFrame* gestisce la terminazione del programma, indica la dimensione di partenza, imposta il menu e visualizza la finestra:

```
private void createUI() {
frame = new JFrame("JAI Photo");
frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
imagePanel = new ZoomableImagePanel();
JPanel content = new JPanel( new BorderLayout() );
JPanel infoBar = createInfoBar();
content.add( infoBar, BorderLayout.SOUTH );
content.add( new JScrollPane( imagePanel ),
                                    BorderLayout.CENTER );
frame.getContentPane().add( content );
//...
frame.setJMenuBar( createMenu() );
frame.setSize( 600, 400 );
frame.setVisible(true); }
```

La barra di stato *(infoBar)* viene costruita con il metodo *createInfoBar()*, che ritorna un *JPanel*, contentente tutti gli elementi richiesti. Ciascuno di questi è racchiuso da un ulteriore pannello, che consente di realizzare effetti particolari; ad esempio, la barra di progressione è spaziata a sinistra da uno *strut* orizzontale di 20 pixel creato con la chiamata *Box.createHorizontalStrut(20)*. Anche la *label* che contiene la dimensione dell'immagine è spaziata sulla destra di 20 pixel, con il medesimo sistema; il combo box che contiene il livello di zoom, invece, dispone di una label alla sua destra che riporta la dicitura %:

```
JPanel createInfoBar() {
  JPanel bar = new JPanel( new BorderLayout() );
  JPanel progressBar1 = new JPanel();
```

```
progressBar = new JProgressBar(
                        JProgressBar.HORIZONTAL, 0, 10 );
progressBar.setValue(0);
progressBar1.add( Box.createHorizontalStrut(20) );
progressBar1.add( progressBar );
JPanel imageSizePanel1 = new JPanel();
imageSizePanel = new JLabel();
imageSizePanel1.add( imageSizePanel );
imageSizePanel1.add( Box.createHorizontalStrut(20) );
JPanel zoomControl1 = new JPanel();
final JComboBox zoomCombobox = new JComboBox(
  new Object[] {"5","10","25","50","75","100","125",
                              "150", "180","200"});
  zoomCombobox.setSelectedItem("100");
  zoomCombobox.addItemListener( new ItemListener() {
    public void itemStateChanged(ItemEvent e) {
        imagePanel.setZoom(Integer.parseInt((String)
              zoomCombobox.getSelectedItem() ) );
        imagePanel.setImage(
              originalImage.getAsBufferedImage());
        refreshUI();}});
    zoomControl1.add( zoomCombobox );
    zoomControl1.add( new JLabel(" %") );
    bar.add( progressBar1, BorderLayout.WEST );
    bar.add( zoomControl1, BorderLayout.CENTER );
    bar.add( imageSizePanel1, BorderLayout.EAST );
  return bar;
}
```

Particolarmente interessante è l'inizializzazione del combo box, che prevede una serie di dimensioni di zoom predefinite, passate direttamente nel costruttore. Con il metodo *setSelectedItem()* viene indicato che lo zoom di default è *100*, mentre il *listener ItemListener* consente di gestire ogni selezione di un nuovo zoom da parte dell'utente. Quando questo avviene, il metodo *itemStateChanged()* viene invocato, ed allora viene estratto l'elemento selezionato utilizzando *getSelectedItem();* il valore viene convertito in intero e passato al pannello di visualizzazione dell'immagine. Per rendere effettiva la modifica dello zoom, viene ripassata l'immagine originale al pannello *immagine*, e viene richiamato il metodo *refreshUI()*, per aggiornare la finestra. Come si vedrà più avanti, il ridimensionamento effettivo avviene alla chiamata di *setImage()* e non alla *setZoom()*.

VISUALIZZARE UNA IMMAGINE

Per visualizzare l'immagine che il programma andrà a manipolare è necessario costruire delle classi apposite, soprattutto considerando che è necessario supportare lo zoom. La classe *ImagePanel* implementa un semplice *Component* che visualizza una immagine di tipo *BufferedImage*. La classe dispone dei seguenti metodi:

- **setImage()** imposta l'immagine da visualizzare;
- **getPreferredSize()** ritorna la dimensione "preferita" del componente, che è solitamente la dimensione dell'immagine stessa;
- **paint()** disegna l'immagine in un contesto *Graphics.*

Si noti che gli ultimi due metodi non sono richiamati dall'applicazione, ma dal sottosistema grafico di Java *(AWT)* quando un componente viene inserito nell'interfaccia utente. Il codice completo è il seguente:

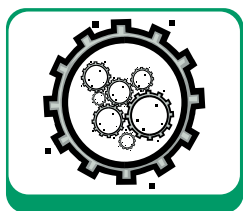
```
package it.bigatti.jaiphoto.ui;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.image.BufferedImage;
public class ImagePanel extends Component {
  BufferedImage image;
  private final Dimension defaultDimension = new
                                        Dimension(500,400);
  public ImagePanel() { }
  public void setImage( BufferedImage image ) {
    this.image = image;}
  public Dimension getPreferredSize() {
    if (image != null) {
      return new Dimension( image.getWidth(),
                                     image.getHeight() );
  } else {
    return super.getPreferredSize();}}
public void paint( Graphics g ) {
  if( image != null ) {
    Graphics2D g2 = (Graphics2D)g;
    g2.drawImage( image, 0, 0, null ); }}
}
```

SCALARE LE IMMAGINI

La classe *ZoomableImagePanel,* che è una sottoclasse di *ImagePanel,* implementa un ridimensionamento dell'immagine utilizzando *JAI.* I metodi presenti in questa classe sono:

- **setImage()** imposta l'immagine su cui operare;
- **setZoom()** imposta il fattore di zoom;
- **zoom()** metodo non pubblico che esegue il ridimensionamento.

Quest'ultimo metodo è quello più importante, anche perché ci introduce alle API *JAI.* Come si noterà, l'utilizzo di queste API è abbastanza semplice; o meglio, è semplice l'interfaccia che offre, perché alcune manipolazioni sono in realtà abbastanza complesse. Per eseguire il ridimensionamento si esegue dunque la chiamata:

Ulteriori informazioni su Java Advanced Imaging sono reperibili in Internet sul sito http://java.sun.com/products/java-media/jai/ dove è possibile accedere ai download delle librerie, di codice aggiuntivo, della documentazione ed accedere a link di articoli correlati alla tecnologia. Sono presenti anche dei business case sui clienti che hanno adottato JAI.

```
PlanarImage pi = JAI.create("scale", params);
```

JAI è la classe principale delle API di Sun relative, e dispone di diversi metodi, incluse alcune versioni di *create()*. Il primo parametro è sempre la tipologia di operazione richiesta: in questo caso è un ridimensionamento, ma si vedrà, nel corso dello sviluppo del resto dell'applicazione, come siano disponibili diverse funzioni. Il secondo parametro è un oggetto di tipo *ParameterBlock*, una sorta di contenitore dentro al quale vengono inseriti tutti i parametri necessari all'elaborazione. Nel caso dell'operazione di ridimensionamento, i parametri passati sono:

- L'immagine di partenza
- Il ridimensionamento orizzontale
- Il ridimensionamento verticale
- La traslazione (spostamento) orizzontale
- La traslazione verticale
- Un oggetto che indica la strategia di ridimensionamento, in questo caso *javax.media.jai.InterpolationNearest*

Fig. 1: La finestra principale dell'applicazione JAIPhoto

Il valore di ritorno di *JAI.create()* è di tipo *RenderedOp*, che sostanzialmente rappresenta una operazione di rendering; la classe *PlanarImage* implementa questo tipo e permette di ottenere una *BufferedImage* attraverso il metodo *getAsBufferedImage()*. Il valore di ritorno viene memorizzato in *image*, che è l'attributo della classe *ImagePanel* che contiene l'immagine da visualizzare. In questo modo, quando AWT richiederà la visualizzazione del pannello, verrà presentata la versione ridimensionata. Il codice completo di *ZoomableImagePanel* è il seguente:

```
package it.bigatti.jaiphoto.ui;
import java.awt.image.BufferedImage;
import java.awt.image.renderable.ParameterBlock;
import javax.media.jai.InterpolationNearest;
import javax.media.jai.JAI;
import javax.media.jai.PlanarImage;
public class ZoomableImagePanel extends ImagePanel {
  float zoom;
  public ZoomableImagePanel() {
    setZoom( 100 ); }
  public void setImage( BufferedImage image ) {
    super.setImage( image );
    zoom(); }
  void zoom() {
    if ( zoom != 1.0f ) {
      ParameterBlock params = new ParameterBlock();
      params.addSource(image);
      params.add(zoom);  //x scale factor
      params.add(zoom);  //y scale factor
      params.add(0.0F);  //x translate
      params.add(0.0F);  //y translate
      params.add(new InterpolationNearest());
      System.out.println("zooming: " + zoom);
      PlanarImage pi = JAI.create("scale", params);
      image = pi.getAsBufferedImage();
      System.out.println("zooming fatto"); } }
  public void setZoom( int perc ) {
    zoom = (float)(perc) / 100f;}
}
```

APPROFONDIMENTI

SUN ha creato un tutorial per JAI che si trova all'indirizzo http://java.sun.com/developer/onlineTraining/javaai/jai/index.html. La struttura è alquanto diversa dai precedenti tutorial realizzati dall'azienda per Java: trattandosi di elaborazioni di immagine, questo tutorial è basato principalmente su applet grafiche di esempio.

CARICAMENTO DELL'IMMAGINE

Il caricamento dell'immagine è implementato nel metodo *load()*, che come parametro una stringa con il nome completo dell'immagine da caricare:

```
void load( final String filename ) {
  runner.run( new JAIOperation() {
    public PlanarImage run() {
      image = JAI.create("fileload", filename);
      originalImage = image;
      return image; }
    public String getDescription() {
      return "loading"; } });
}
```

Il metodo è piuttosto semplice, ed utilizza ancora una volta il metodo *JAI.create()*, questa volta indicando la funzione *fileload* e passando come parametro il nome del file. Il metodo ritorna una immagine che viene memorizzata nell'attributo *image*. Si noti che *load()* è un metodo di *Main*, mentre *image* e *originalImage* (l'assegnazione successiva) sono attributi di questa classe, e non di *ImagePanel*. Il metodo utilizza due oggetti sconosciuti: *JAIOperation* e *runner*. Il primo è una semplice interfaccia che serve ad incapsulare una operazione sull'immagine:

```
package it.bigatti.jaiphoto.utils;
import javax.media.jai.PlanarImage;
public interface JAIOperation {
  PlanarImage run();
  String getDescription();
}
```

Il secondo oggetto, *runner*, è di tipo *JAIRunner*, ed entrambi non sono altro che classi di supporto definite nel package *it.bigatti.jaiphoto.utils* e che consentono di eseguire le operazioni di manipolazione delle immagini, caricamento e quant'altro in modo asincrono rispetto al thread principale del programma. In questo modo, l'interfaccia utente resta legge-

ra e veloce, e può visualizzare feedback su quanto sta avvenendo dietro le quinte. Si consideri, infatti, che JAI consente la manipolazione di immagini anche di grosse dimensioni, come quelle satellitari o per la scienza medica, elaborazioni che potrebbero impiegare un discreto tempo di CPU. L'oggetto *runner*, attributo della classe *Main*, viene creato nel metodo *createUI()* (ma è stato omesso nel listato precedente per non creare confusione), in questo modo:

```
runner = new JAIRunner( new UICallback() {
  public JProgressBar getProgressBar() {
    return progressBar; }
  public ImagePanel getImagePanel() {
    return imagePanel; }
  public JLabel getImageSizePanel() {
    return imageSizePanel; }
  public void refresh() {
    refreshUI(); }
  public PlanarImage getImage() {
    return image; }}
);
```

La classe *JAIRunner* è strutturata molto semplicemente con due metodi:

- **run()** esegue una operazione di tipo *JAIOperation* all'interno di un nuovo thread;
- **getImage()** ritorna il risultato dell'elaborazione dell'operazione eseguita.

La parte più interessante è sicuramente contenuta nel metodo *run()*. Qui viene creato un nuovo oggetto *Thread*, implementando nel suo metodo *run()* una serie di attività:

- stampa la descrizione dell'operazione in corso;
- viene eseguita l'operazione in corso, chiamando *run()* sull'oggetto *JAIOperation* passato;
- se l'immagine risultante non è nulla, viene passata alla finestra principale attraverso l'oggetto di callback e viene aggiornata l'interfaccia utente (*label* con la dimensione dell'immagine);
- viene effettuato il refresh della finestra
- stampa nuovemente l'operazione attuale.

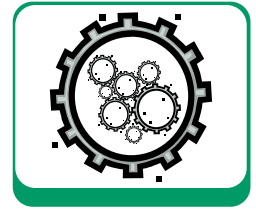
In mezzo a tutte queste operazioni, è impostata la barra di scorrimento sulla modalità "indeterminato". Non è infatti possibile prevedere quanto tempo richiede un'operazione di manipolazione, ma in questo modo viene fornito comunque un feed-back all'utente. Una barra indeterminata visualizza un generico segnale di progressione, ma non è in grado di rendere l'idea della quantità di tempo mancante al completamento dell'operazione. Il codice di *JAIRunner* è il seguente:

```
package it.bigatti.jaiphoto.utils;
import it.bigatti.jaiphoto.ui.UICallback;
import javax.media.jai.PlanarImage;
public class JAIRunner {
  PlanarImage image;
  UICallback callback;
  public JAIRunner( UICallback callback ) {
      this.callback = callback; }
   public void run( final JAIOperation operation ) {
     callback.getProgressBar().setIndeterminate(true);
     Thread thread = new Thread() {
        public void run() {
          System.out.println( operation.getDescription() );
          //esegue l'operazione
          image = operation.run();
          //visualizza l'immagine di ritorno
          if (image != null) {
              callback.getImagePanel().setImage(
                         image.getAsBufferedImage() );
              //imposta la descrizione con la dimensione
              callback.getImageSizePanel().setText(
                          image.getWidth() + " x "
                              +image.getHeight() );
          } else {
              callback.getImageSizePanel().setText("N/A"); }
          //aggiorna la finestra
          callback.refresh();
          System.out.println( operation.getDescription()
                                     + " fatto" );
        callback.getProgressBar().setIndeterminate(false);
     } };
    thread.start(); }
  public PlanarImage getImage() {
    return callback.getImage(); }}
```

Utilizzando questa tecnica, l'interfaccia utente e *JAIRunner* sono disaccoppiati e quest'ultimo potrebbe essere riutilizzato in un'altra applicazione.
Il codice di *UICallback* è il seguente:

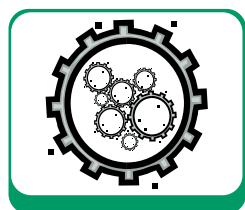
```
package it.bigatti.jaiphoto.ui;
import javax.media.jai.PlanarImage;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JProgressBar;
public interface UICallback {
  JProgressBar getProgressBar();
  ImagePanel getImagePanel();
  JLabel getImageSizePanel();
  PlanarImage getImage();
  void refresh();}
```

La lettura del codice evidenzia che, da un oggetto *UICallback* è possibile ottenere la barra di avanzamento, il pannello immagine, la label che riporta la dimensione dell'immagine e l'immagine stessa. Inoltre, è presente il metodo *refresh()*, che aggiorna tutta la finestra dell'applicazione.

Massimiliano Bigatti

I *callback* sono chiamate che un oggetto figlio restituisce all'oggetto padre per notificarlo di avvenimenti di cui potrebbe essere interessato. Nel linguaggio Java si può realizzare questa funzionalità anche con i *listener*.

Sfruttiamo WSE2, DIME e .NET per mandare file con un IM

Instant Messenger personalizzato parte seconda

In questo articolo diremo qualcosa sui formati usati dai web services per la gestione dei file attachment, gestiremo il Drag&Drop, invieremo un messaggio e impareremo come riceverlo

Utilizza questo spazio per le tue annotazioni

I software attuali ci hanno abituato all'utilizzo degli allegati. Inviare un file da un capo all'altro del mondo è ormai una prassi standard. Generalmente, però, chi riceve il nostro messaggio con l'allegato è una persona che, sul momento, decide cosa fare (aprire, salvare o cancellare il nostro file). Cosa succede se dall'altro lato non c'è un essere umano? Peggio ancora, come dobbiamo comportarci se il sistema che riceve il messaggio è diverso dal nostro? Fortunatamente esistono degli standard a cui i produttori di software devono attenersi. I Web Service si attengono ad una serie di specifiche denominate *WS-**. Una di queste, *WS-Attachments,* descrive la maniera standard in cui inviare un allegato incapsulandolo in un messaggio SOAP. Se il sistema adotta il medesimo standard, non ci saranno problemi di comunicazione e sia il messaggio sia l'allegato verranno gestiti correttamente. Si parla, in questi casi, di *interoperabilità*. Nel nostro caso particolare, stiamo utilizzando un software scritto in C# su piattaforma Microsoft, quindi non risentiamo in modo particolare del problema dell'interoperabilità. Utilizzeremo come standard per la trasmissione degli allegati quello denominato DIME di cui parleremo fra un attimo.

Fig. 1: L'Instant Messenger che realizzeremo

ANATOMIA DI UN ATTACHMENT

Le specifiche *WS-Attachments* descrivono i metodi standard in cui aggiungere allegati ai messaggi SOAP. Innanzitutto, quando abbiamo la necessità di creare un *DIME Attachment,* il nostro messaggio viene creato da un *DIME Generator* e ricevuto da un *DIME Parser.* Il *DIME Generator* si occupa di creare un messaggio composto da uno o più *DIME record* che si "autodescrivono". Per comprendere meglio i passaggi sopraelencati, vediamo cosa accade alla ricezione di un messaggio DIME. La **Figura 2** mostra la struttura di un singolo record DIME. Generalmente, un record rappresenta il nostro allegato. Come abbiamo detto in precedenza, possiamo aggiungere più allegati ad un messaggio, il che significa che il *DIME Generator* accoderà via via tutti i record necessari. In presenza di allegati di dimensioni considerevoli, i nostri record potrebbero anche rappresentare dei frammenti di un unico allegato (lo vedremo più avanti). Detta così potrebbe sembrare di difficile comprensione ma, alla fine di questo paragrafo, vi assicuro che avrete le idee più chiare. Riprendiamo il nostro record e la **Figura 2**. Quando un messaggio DIME viene ricevuto dal *DIME Parser,* i record in esso contenuti vengono analizzati in modo sequenziale. Ogni record contiene un *Header* (parte alta della figura) oltre ai dati veri e propri (parte bassa).

L'*header* a sua volta è diviso in svariati blocchi. Il parser inizia la lettura dalla *Version* e verifica se è in grado di interpretare il messaggio. In caso di versione non compatibile, il messaggio viene rifiutato. Se il ritorno è positivo invece, il parser passa alla lettura del *Message Begin* (inizio del messaggio). Questo flag è settato a 1 (vero) solo nel primo record. In caso di assenza di tale flag nel primo record o di presenza nei record successivi, il messaggio intero viene ritenuto "non valido" e viene rifiutato. Si passa poi alla lettura del *Message End* (fine del messaggio). Al contrario del *Message Begin*, il *Message End* deve essere presente solo nell'ultimo record. Nel caso di un unico record, *Message Begin* e *Message End* saranno entrambi veri. Se non vengono riscontrati problemi nella lettura di *Message Begin* e *Message End*, si passa al *Chunk Flag*. Questo flag merita una piccola parentesi. In precedenza abbiamo detto che un record può rappresentare un allegato o una parte di esso. In caso di allegati di grosse dimensioni, questo viene "spezzato" *(Chunked)* ed ogni pezzo viene inserito come record nel messaggio. Spezzare un allegato di grosse dimensioni conviene sostanzialmente per due motivi:

1. Ottimizzazione delle prestazioni dovute al minor carico di memoria necessaria a bufferizzare grossi file.

2. Superamento del limite di dimensioni di ogni singolo record all'interno del messaggio (lo vedremo più avanti).

Il *Chunk Flag* funziona in questo modo: è settato ad 1 (vero) per tutti i record dello stesso allegato, tranne l'ultimo. Il parser, in presenza di un *Chunk Flag* settato a vero, assume che tutti i record successivi che hanno lo stesso flag rappresentano un unico allegato. Il primo record con *Chunk Flag* a 0 è l'ultimo "pezzo" dell'allegato. I record successivi vengono considerati con lo stesso principio. I dati contenuti nel *TYPE_T* verranno utilizzati dal parser per determinare come leggere il *TYPE* che incontrerà successivamente. *RESERVED* viene saltato. Esiste solo in previsione di nuove versioni di *DIME*. La lettura dell'Header prosegue dunque con *OPTIONS_Lenght, ID_Lenght, TYPE_Lenght* e *DATA_Lenght*. Questi campi contengono le dimensioni dei rispettivi campi relativi ai dati (subito in basso nell'Header). I primi campi *(OPTION, ID, TYPE)* possono contenere un massimo di 64KB di dati. *DATA* invece può contenere un massimo di 4GB. Ed è qui che diventa evidente il vantaggio della suddivisione di un

Fig. 2: **Anatomia di un DIME record**

grosso allegato in più pezzi: il limite di 4GB è relativo ad ogni singolo record. Tale limite è definito nel campo *DATA_Lenght* che è un unsigned 32bit integer. Non esiste però (almeno teoricamente) un limite al numero di record inseribili in un messaggio *DIME*. Dopo aver letto i campi relativi alle lunghezze degli elementi del record, il parser legge il campo *OPTIONS* che fornisce una serie di dati extra. Se il parser non li riconosce o il campo è vuoto, viene semplicemente ignorato. Lo step successivo è la lettura dell'ID che specifica un URI che identifica in modo univoco il record. Il *TYPE* identifica il tipo di dato contenuto nel record e, in base alle informazioni recuperate nel campo *TYPE_T*, questo è riferito in base ad un URI o ad un *MIME-Type*. Il campo finale è il *DATA* che finalmente contiene il nostro allegato. Tutti gli elementi letti dal parser servono, alla fine, ad estrarre in modo corretto l'allegato, a ricostruirlo, se necessario, ed a salvarlo sul nostro Hard Disk. Per un rapido riferimento ai significati dei vari campi, fare riferimento alla **Tabella 1**.

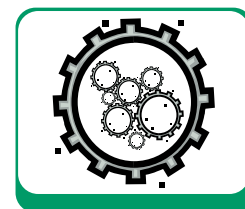
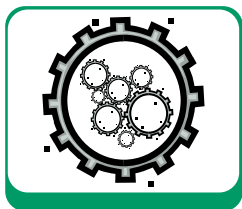



Fig. 3: **Message Begin e Message End Flags**

Campo	Descrizione
VERSION (5 bit)	Versione *DIME*
MB (1 bit)	Flag che specifica se il record corrente è il primo
ME (1 bit)	Specifica se il record corrente è l'ultimo
CF (1 bit)	Specifica se il record è una porzione di un file "spezzato" (*chunked*)
TYPE_T (4 bit)	Specifica la struttura ed il formato del campo *TYPE*
RESERVED (4 bit)	Riservato per usi futuri
OPTIONS_LENGTH (16 bit)	Specifica la lunghezza in bytes del campo *OPTION*
ID_LENGTH (16 bit)	Specifica la lunghezza in bytes del campo *ID.*
TYPE_LENGTH (16 bit)	Specifica la lunghezza in bytes del campo *TYPE*
DATA_LENGTH (32 bit)	Specifica la lunghezza in bytes del campo *DATA*
OPTIONS	Contiene ogni informazione opzionale usata dal *DIME Parser*
ID	Contiene un URI per l'identificazione univoca dell'allegato. La lunghezza è specificata dal campo *ID_LENGHT*
TYPE	Specifica la codifica per il record
DATA	Contiene l'allegato

TABELLA 1: Sintesi dei dati contenuti nel record DIME

INVIAMO IL PRIMO MESSAGGIO

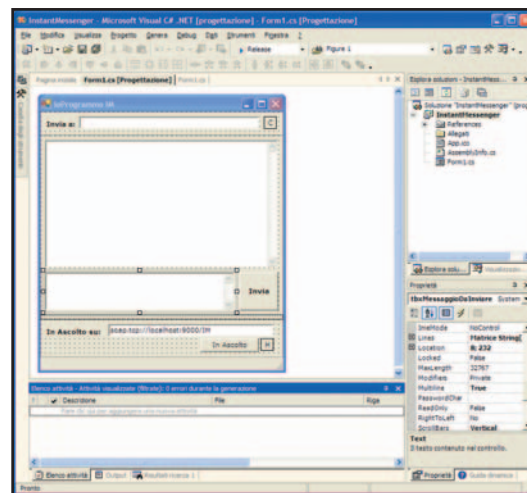
Dopo aver analizzato in dettaglio la struttura di un messaggio DIME ed aver compreso quali sono i passaggi che vengono effettuati internamente, realizziamo qualcosa di concreto. Ampliamo quindi l'Instant Messenger visto nel numero precedente aggiungendo il supporto per l'invio di file. Come prima cosa, apportiamo una piccola modifica all'interfaccia (sostanzialmente invariata rispetto all'articolo precedente), per rendere agevole l'invio di file. I programmi professionali consentono l'aggiunta di elementi tramite una procedura denominata *Drag and Drop*. In sostanza si tratta di trascinare un elemento esterno all'interno di un programma che si occuperà di elaborarlo. Tale comportamento va abilitato all'interno del programma e, ovviamente, va gestito. L'elemento su cui abilitare il *Drag and Drop* è l'area in cui scriviamo i messaggi da inviare quindi, una volta selezionato visualizziamo la scheda *Eventi dell'editor* (se usiamo Visual Studio). Associati gli eventi a cui fare riferimento quando un elemento viene trascinato, spostiamoci nel codice e definiamone il comportamento.

Un articolo completo (in inglese) a cui fare riferimento per maggiori dettagli su DIME è: http://msdn.microsoft.com/msdnmag/issues/02/12/DIME di Jeannine Hall Gailey.

Fig. 4: ***L'interfaccia utente***

```
tbxMessaggioDaInviare_DragEnter(...)
```

scatta quando l'elemento che vogliamo inviare viene trascinato nel controllo associato ma il pulsante del mouse non è ancora stato rilasciato.

Lo scopo di questo evento è sostanzialmente quello di intercettare il tipo di allegato e abilitare un comportamento specifico. Nel nostro caso, qualsiasi sia il file "trascinato" sul controllo, ne abilitiamo la copia. Lo step successivo è quello di elaborare, all'interno del programma, il file vero e proprio:

```
private void tbxMessaggioDaInviare_DragDrop(object
```

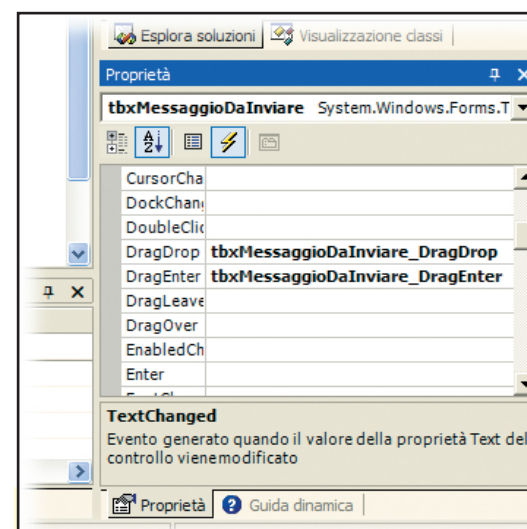



Fig. 5: ***Visualizzazione degli eventi relativi alla textBox***

GLI ALTRI STANDARD

Su sistemi più complessi, di tipo enterprise, l'interoperabilità è un problema molto sentito e avere a disposizione degli strumenti standard diventa estremamente comodo. Ad oggi esistono tre modi per aggiungere un allegato ad un messaggio SOAP e sono:

- **Codifica Base64: il file da allegare viene codificato in base64 (il framework .NET prevede già delle classi per farlo). Il risultato della codifica può essere poi allegato al messaggio, ed essere decodificato al suo arrivo.**
- **DIME Direct Internet Message Incapsulation. DIME è implementato in WSE1 e WSE2 e ha dalla sua parte una grande semplicità di utilizzo. Ne parliamo in questo articolo.**
- **MTOM Message Transmission Optimization Mechanism. È il futuro: verrà utilizzato da Indigo e, attualmente, non è supportato da nessun toolkit.**

Vediamo brevemente quali sono i vantaggi e gli svantaggi di questi sistemi. Il primo sistema (base64) è forse il più immediato ma è poco pulito. Non fa infatti distinzione tra messaggio vero e proprio e allegato. Il risultato dell'encoding, infatti, viene inserito in un campo del messaggio stesso. Utilizzare la codifica Base64 è un buon sistema quando dobbiamo trattare allegati di piccole dimensioni, ma si rivela inefficiente di fronte a sostanziose moli di dati. Il secondo sistema (DIME) è, ad oggi, ampiamente supportato e facile da utilizzare ma ha uno svantaggio abbastanza pesante: sparirà per far posto a MTOM. Allo stato attuale, il supporto è garantito per almeno cinque anni, ma c'è la certezza che con Indigo non verrà più utilizzato. Il grosso vantaggio di DIME è che consente di gestire in maniera abbastanza semplice anche allegati di grosse dimensioni. Il terzo sistema (MTOM), come già detto, sarà il futuro ma ad oggi è ancora in fase di definizione. È in fase di approvazione da parte del W3C e non è ancora arrivato alla fase di Raccomandation. Usarlo, quindi, oltre a non essere semplicissimo (non è ancora supportato dal framework .NET), è anche rischioso perché tra i draft (documenti descrittivi) e la Raccomandation del W3C (che definisce lo standard), potrebbero cambiare diverse cose. Quale scegliere allora? Scartato MTOM, la scelta deve essere fatta tra la codifica base64 e DIME.

```
        sender, System.Windows.Forms.DragEventArgs e)
{
    string file = ((string[])e.Data.GetData(
                                DataFormats.FileDrop))[0];
    string ext;
    string FileName = ExtractFileName(file, out ext);
    InsertImage(file, ext);
    rtbMessaggi.AppendText("Invio del file " + FileName
                                                +"\r\n");
    this.Filename = FileName;
    SendAttachment(file);
}
```

Le prime sei righe di questo metodo servono a visualizzare il messaggio da inviare al destinatario insieme all'allegato. L'allegato vero e proprio viene inserito nel messaggio *SOAP* dal metodo *SendAttachment*:

```
using Microsoft.Web.Services2.Dime;
.
.
private void SendAttachment(string file){
    messaggio = new SoapEnvelope();
    messaggio.SetBodyObject(tbxMessaggioDaInviare.Text);
    messaggio.Context.Addressing.Action = new
                                        Action("urn:IM");
    messaggio.Context.Addressing.From = new From(
                        new Uri(tbxInAscoltoSu.Text));
    messaggio.Context.Addressing.To = new To(new Uri(
                                        tbxTo.Text));
    //Creazione dell'allegato DIME
    DimeAttachment da = new DimeAttachment(
                "image/gif", TypeFormat.Unknown, file);
    da.Id = this.Filename;
    messaggio.Context.Attachments.Add(da);
    messaggio.SetBodyObject(tbxMessaggioDaInviare.Text);
    SoapSender sSender = new SoapSender(new
                                    Uri(tbxTo.Text));
    sSender.Send(messaggio);
}
```

La costruzione dell'allegato avviene creando una istanza della classe *DimeAttachment*. Sono presenti ben cinque overload del costruttore, il che ci dà abbastanza flessibilità nella costruzione del messaggio (**Figura 6**). Nel nostro caso si è scelto di utilizzare il terzo, che richiede in ingresso rispettivamente una stringa che identifica il tipo di allegato che stiamo inviando, un *TypeFormat* per il formato e il percorso del file da allegare:

```
DimeAttachment da = new DimeAttachment(
                "image/gif", TypeFormat.Unknown, file);
da.Id = this.Filename;
```

Il campo *ID*, come abbiamo visto nel precedente paragrafo, serve a identificare in modo univoco l'allegato. Generalmente, a tale campo va associato un *GUID* che garantisce l'univocità. Nel nostro caso, dato che la leggibilità è prioritaria rispetto all'univocità, usiamo tale campo per passare il nome del file. L'allegato *DIME (da)* viene aggiunto all'*envelope (messaggio)* con l'istruzione:

```
messaggio.Context.Attachments.Add(da);
```

e, proprio come un normalissimo messaggio (vedi articolo precedente), viene inviato al destinatario:

```
SoapSender sSender = new SoapSender(new
                                    Uri(tbxTo.Text));
sSender.Send(messaggio);
```

LA RICEZIONE

Alla ricezione del messaggio, dobbiamo verificare l'eventuale presenza di allegati da estrarre e da salvare sull'Hard Disk. L'allegato, infatti, sarà parte integrante del messaggio *SOAP* ricevuto:

```
protected override void Receive(SoapEnvelope envelope)
{
    MessaggioRicevuto.AppendText(
                    envelope.GetBodyObject(typeof(String)
                                    ).ToString()+"\r\n");
    if ( envelope.Context.Attachments.Count == 1 )
    {
```

Se il conteggio degli allegati è uguale ad 1, possiamo procedere alla sua estrazione.
È importante notare che, nel caso specifico dell'*Instant Messenger* in cui possiamo inviare un allegato alla volta, controllare che il conteggio degli allegati è uguale ad 1 è corretto ma, come abbiamo visto in precedenza, potendo allegare più elementi allo stesso messaggio, dobbiamo tenerne conto in sviluppi futuri:

```
SalvaAllegato(envelope.Context.Attachments[0]);
MessaggioRicevuto.AppendText("L'allegato "
        + envelope.Context.Attachments[0].Id + " è stato
                    salvato nella directory Allegati\r\n");
```

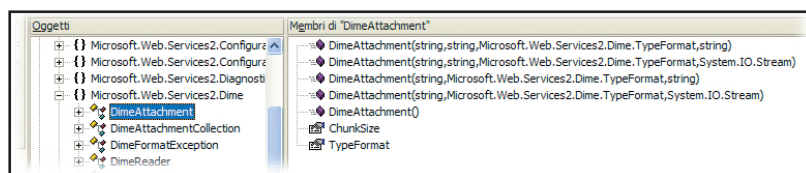
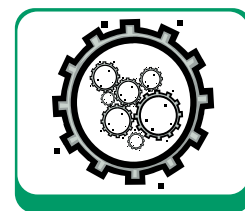
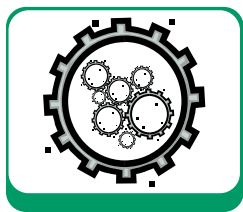



***Fig. 6:** Visualizzazione metodi DimeAttachment*

Recentemente è uscito WSE 2 SP1 che introduce miglioramenti a WSE 2.
Da tenere sott'occhio è sicuramente il sito http://msdn.microsoft.com/webservices/ **in cui vengono riportate tutte le informazioni relative al mondo dei WS.**

Le specifiche su *WS-Attachments* sono disponibili, insieme a tutte le altre, a questo indirizzo: http://msdn.microsoft.com/webservices/understanding/specs/default.aspx

Appurata la presenza del messaggio, lo passiamo come argomento al metodo *SalvaAllegato* che si occuperà di estrarlo e di salvarlo nella nostra cartella predefinita:

```
private void SalvaAllegato(Microsoft.Web.
            Services2.Attachments.Attachment allegato)
{
  System.IO.Stream s = allegato.Stream;
  Byte[] b = new byte[s.Length];
  s.Read(b, 0, b.Length);
  s.Close();
  System.IO.FileStream f = System.IO.File.OpenWrite(
                        Environment.CurrentDirectory+
                          "\\Allegati\\"+allegato.Id);
  f.Write(b, 0, b.Length);
  f.Close();
}
```

In definitiva, *allegato.Stream* crea uno stream di byte a partire dall'attachment. Il *DIME Parser* segue passo passo le operazioni elencate nel secondo paragrafo ed estrae, sotto forma di stream, i dati dall'envelope.
Lo stesso stream viene poi passato alle istruzioni

```
System.IO.FileStream f = System.IO.File.OpenWrite(
                        Environment.CurrentDirectory+
                          "\\Allegati\\"+allegato.Id);
f.Write(b, 0, b.Length);
```

che hanno il compito di ricostruire sul file sull'Hard Disk.

INSTALLAZIONE E USO

Per il corretto funzionamento dell'Instant Messenger è necessario installare sul PC sia il Microsoft .NET Framework che il runtime di WSE 2 (nel primo box laterale il link per il dowload). Una volta installati questi due elementi, siamo pronti a farlo funzionare. Per un rapido test possiamo avviare due volte il programma e predisporlo alla comunicazione. Nella prima finestra, dichiariamo un indirizzo, completo di porta, in cui metterci in ascolto. Ad esempio *soap.tcp://localhost:9000/IM* e clicchiamo sul bottone *In Ascolto*. Nella seconda finestra, definiamo una porta diversa, ad esempio *soap.tcp://localhost:9001/ IM* e, anche qui, clicchiamo sul relativo bottone. Torniamo nella prima finestra, inseriamo l'indirizzo di destinazione nella casella *To* (in questo caso sarà quello in cui abbiamo messo in ascolto la seconda finestra), scriviamo qualcosa nell'apposita area e premiamo *Invia*. Il nostro messaggio sarà inviato all'indirizzo del destinatario e, oltre ad apparire nell'area dedicata ai messaggi, verrà compilato automaticamente anche il campo *To*.
Se il test va a buon fine, possiamo utilizzare il nostro *IM* su *due pc* diversi. La procedura è identica tranne che per due fattori: l'indirizzo di ascolto *(localhost)* deve essere sostituito con l'IP pubblico del PC su cui è in esecuzione il programma. Per recuperarlo, avviamo la shell *(start/esegui/cmd)* e digitiamo il comando *ipconfig*. Se inoltre abbiamo un firewall attivo (Windows XP lo implemente), dobbiamo aprire la porta scelta per la ricezione dei messaggi (nel caso dell'esempio è la porta 9000). Per inviare un file ci basta sceglierne uno dal nostro Hard Disk e trascinarlo nella area di scrittura dei messaggi.
Il file verrà automaticamente inviato al destinatario e verrà salvato nella cartella *Allegati*, presente nella stessa cartella di esecuzione del programma.

SUL WEB

Per chi fosse interessato a MTOM, sul sito del W3C sono presenti le specifiche ufficiali all'indirizzo www.w3.org/TR/2004/CR-soap12-mtom-20040826/

NOTA

INDIGO

Nella prossima release di Windows, denominata Longhorn, avremo un nuovo modello, costituito da un set di API e denominato Indigo. Indigo, alla cui realizzazione sta lavorando attivamente anche Don Box, offrirà servizi e funzioni per la gestione della comunicazione fra applicazioni, tramite lo scambio e la gestione dei messaggi, (accodamento, instradamento) sfruttando i protocolli più comuni, quali: TCP, HTTP, IPC, ecc.
Indigo nasce come evoluzione della tecnologia Web Services, integrando funzioni evolute per la sicurezza e la gestione delle transizioni. Lo stesso Indigo, dovrebbe essere il cuore pulsante per lo sviluppo di nuovi tools di collaborazione della suite Office; ciò consentirà, a più persone, di lavoreare contemporaneamente al medesimo progetto Power Point, Excel, ecc.

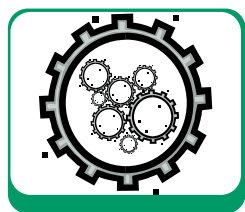
CONCLUSIONI

In questi due articoli abbiamo dato uno sguardo alla tecnologia *Web Service Enancements*, giunti alla versione due.
Abbiamo focalizzato la nostra attenzione su uno degli aspetti più affascinanti di questa tecnologia: l'assenza di un server web. WSE introduce molte novità rispetto ai classici Web Service ed una applicazione di tipo peer to peer esalta alcune di queste caratteristiche.
Si parla spesso di WSE come la strada verso *INDIGO* e l'architettura Service Oriented e, in questo momento di transizione, aggiornarsi è un obbligo. Buon lavoro.

Michele Locuratolo

KSOAP e J2ME i Web Services approdano sui dispositivi mobili

Il meteo nel cellulare

In questo articolo ci colleghiamo ad un Web Service, grazie al quale recuperiamo le informazioni meteorologiche dal web. Faremo funzionare il tutto dal telefonino con la libreria KSOAP

Utilizza questo spazio per le tue annotazioni

J2ME, ovvero, l'insieme di specifiche utilizzato da java per la creazioni di applicazioni rivolte ai dispositivi mobili, sta diventando sempre più potente e completo. In questo senso è interessante legare l'insieme di potenzialità associate all'uso, per esempio, di un cellulare, con quelle offerte da un Web Service. Senza entrare troppo nel dettaglio, è utile ricordare che un Web Service è una qualche applicazione residente in rete e che svolge un compito particolare. Ad esempio è possibile interrogare un Web Service che serve a conoscere le previsioni meteo, e inserire la risposta in una nostra applicazione. Se la nostra applicazione gira in un telefonino tutto divente particolarmente interessante. Inutile dire che non c'è niente che richieda un intervento umano nell'effettuare la richiesta al Web Service. Sarà la nostra applicazione che lo interrogherà usando il protocollo SOAP e allo stesso modo ne decifrerà la risposta. Ugualmente è superfluo dire che non ci interessa conoscere i dettagli di come il Web Service recupererà l'informazione che ci serve. A noi interessa solo avere una risposta.

SUN E I WEB SERVICES

La Sun Microsystems ha già pensato ad un package, WSA, che permette di utilizzare dei Web Service dalla piattaforma J2ME. Purtroppo, per utilizzare questo package aggiuntivo il nostro dispositivo deve avere almeno la versione 2.0 di MIDP *(Mobile Information Device Profile)*, profilo che definisce il ciclo di vita dell'applicazione, gli eventi e l'interazione con l'utente, che attualmente non è disponibile sulla maggioranza dei cellulari. Per poter avere un'applicazione che sicuramente sarà portabile su tutti i dispositivi J2ME dobbiamo utilizzare un libreria esterna compatibile con MIDP 1.0: ecco che entra in campo *kSOAP.*

LIBRERIE KSOAP

kSOAP è una libreria che implementa parzialmente le specifiche 1.1 di SOAP proprio perché è stata pensata principalmente per essere utilizzata su dispositivi mobili che supportano J2ME.
Sviluppata inizialmente da *Enhydra.org*, kSOAP è arrivata alla versione 2.0 ed è un progetto portato avanti dalla Kobject. Queste librerie si basano su un altro importante package utilizzabile sui dispositivi J2ME, ovvero kXML, una libreria che permette di effettuare il parsing XML.
Basandosi appunto su questo package è stato sviluppato un completo insieme che permette una semplice comunicazione con i tanti Web Services esistenti al mondo.
kSOAP è gratuitamente scaricabile dal sito di Enhydra (*http://ksoap.objectweb.org/software/downloads/index.html*), dove troviamo attualmente la versione 2.0.

DALLA TEORIA ALLA PRATICA

Addentriamoci ora nelle API di kSOAP per conoscere le classi e i metodi che andremo a utilizzare nello sviluppo del nostro programma. Quando abbiamo a disposizione un Web Service che vogliamo interrogare, dobbiamo prima di tutto avere informazioni come il namespace, l'URL del servizio e la *SOAP Action*.
Una volta ottenute queste informazioni, possiamo iniziare a studiare il Web Service che vogliamo interrogare. SOAP permette di avere due diversi tipi di interazione, *CALL* e *RESPONSE.*
Un'interazione *CALL* è quella che avviene quando richiamiamo il servizio remoto. *RESPONSE* invece è semplicemente il risultato dell'elaborazione precedentemente richiesta. Avremo dunque biso-

gno di capire quali sono i metodi remoti che possiamo richiamare e le risposte che otterremo, con i relativi argomenti di input e di output. Per il Web Service che andremo a utilizzare nel nostro esempio tutte le informazioni che ci servono sono documentate all'indirizzo *http://www.capescience.com/webservices/airportweather/index.shtml*.
La seconda informazione da conoscere è che all'interno delle librerie *kSOAP* ci sono due classi principali che saranno utili per usufruire di un servizio remoto *SoapObject* e *HttpTransport*.
SoapObject serve per costruire delle chiamate SOAP e, grazie ai metodi presenti all'interno di questa classe, possiamo aggiungere, settare e recuperare le proprietà del nostro oggetto. *HttpTransport* serve, invece, per instaurare la connessione con il server che ospita il WebService ed effettuare delle interazioni di tipo *CALL* e *RESPONSE*.

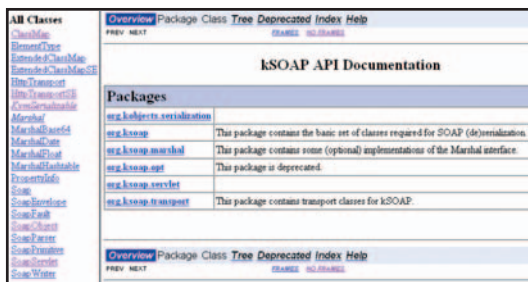



Fig. 3: ***API a disposizione all'interno delle librerie kSOAP***

IL SOLE SULL'AEROPORTO

AirportWeather è un Web Service gratuitamente disponibile che permette di conoscere informazioni riguardanti le condizioni atmosferiche di tutti gli aeroporti che hanno un numero *ICAO (International Civil Aviation Organization)*, quasi tutti in qualsiasi parte del mondo. Abbiamo a disposizione diversi metodi per ottenere la temperatura, il vento, l'umidità e altre informazioni e inoltre anche un metodo che permette di avere tutte le informazioni nella risposta che viene mandata dal server. Consultando la documentazione relativa al Web Service all'indirizzo *http://www.capescience.com/webservices/airportweather/index.shtml* e aiutandoci con la sua definizione formale WSDL all'indirizzo *http://live.capescience.com/wsdl/Airport Weather.wsdl* scopriamo quanto segue:

```
<?xml version="1.0" encoding="UTF-8" ?>
<definitions name="AirportWeather"
        targetNamespace=http://www.capeclear.com/
                AirportWeather.wsdl xmlns=[...]>
[...]
  <message name="getHumidity">
  <part name="arg0" type="xsd:string" />
  </message>
  <message name="getTemperature">
[...]
```

e cioè che il Web Service in questione supporta un metodo *getHumidity* e il metodo *getTemperature* che ricevono in input una stringa che altro non è che il codice ICAO di cui vogliamo conoscere le condizioni meteo e restituiscono rispettivamente umidità e temperatura in sede all'aeroporto richisto.
È utile sapere che il Web Service supporta molti altri metodi tra cui *GetSummary* che ci ritorna informazioni come: umidità, posizione, pressione, condizioni del cielo, temperatura, visibilità e velocità del vento.
Per ciò quello che dobbiamo fare è invocare il Web Service con una chiamata *Call* e decifrarne la risposta.

Fig. 2: ***Passi base per accedere a dei WebService attraverso WSDL e UDDI.***

DEFINIZIONE DELL'INTERFACCIA

Partendo dalla impostazione della interfaccia, vedremo come implementare un piccolo client J2ME che ci permetta di usare le librerie kSOAP per utilizzare il servizio Web appena descritto.

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;
import java.io.*;
import javax.microedition.io.*;

import org.ksoap.*;
import org.ksoap.transport.*;
import org.kxml.*;
import org.kxml.io.*;
import org.kxml.parser.*
```

1 GLI STRUMENTI
Prima di tutto importiamo le librerie che ci servono. Da notare che abbiamo importato ksoap, che implementa le specifiche di comunicazione con il web service

NOTA

CHE COSA È UNA MIDLET?

Le Midlet, con buona approssimazione, sono applicazioni scritte per funzionare su dispositivi J2ME MIDP. Si compongono di due file. Il file *.jad* contiene un breve testo che descrive l'applicazione. Il file *.jar* contiene l'applicazione vera e propria.

CHE COSA È UNA MIDP?

È uno standard che descrive i profili disponibili per dispositivi portatili come i cellulari.

AIRPORT-WEATHER

È un web service messo a disposizione dalla *CapeScience.com*, una comunità di sviluppatori di web service. Sul sito troviamo a disposizione anche molti altri servizi interessanti con una spiegazione dettagliata di ogni Web Service e un'interfaccia Web per testarlo direttamente.

PROVARE WSA

Il package aggiuntivo WSA è consultabile nella lista di JSR approvate http://jcp.org/en/jsr/detail?id=172, dove possiamo inoltre capire il percorso che ha fatto questa specifica e trovare tante altre JSR in attesa di approvazione, specialmente per quanto riguarda il mondo J2ME

TRASMETTERE LE IMMAGINI

Una importante feature delle librerie kSOAP è la possibilità di inviare e ricevere array di byte (ad esempio un'immagine) tramite le richieste e le risposte SOAP. Questo è possibile grazie ad un encoder/decoder Base64 che è stato inserito nelle librerie e che, tramite la costruzione di un oggetto primitivo *SoapPrimitive*, può essere inserito all'interno di una richiesta SOAP o essere ricevuto dall'applicazione sul dispositivo J2ME come mostrato di seguito:

```
SoapPrimitive sp=new
SoapPrimitive (Soap.ENC,
"base64", Base64.encode
(ArrayDiByte));
```

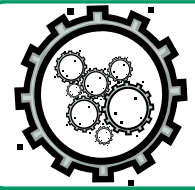
```
public MeteoWS()
{
  codiciAeroporti = new String[] {
  "LIRF", "LIML", "LIRN", "LILI", "LIME", "Limp"};
  aeroporti = new String[] {
     "Roma Fiumicino", "Milano Linate ", "Napoli",
                 "Vercelli", "Bergamo", "Parma" };
  servizi = new String[] {
     "Tutte", "Umidità", "Pressione", "Condizioni
            del cielo", "Temperatura", "Visibilitità",
                         "Velocità del vento" };
  airportMenu = new List( "Scegli un aeroporto",
                    List.IMPLICIT, aeroporti, null );
  serviziMenu = new List( "Scegli un servizio",
                      List.IMPLICIT, servizi, null );
  response = new Alert( "Meteo", null, null,
                                   AlertType.INFO);
  response.setTimeout( Alert.FOREVER );
  backCommand = new Command( "Indietro",
                              Command.BACK, 2 );
  exitCommand = new Command( "Esci",
                              Command.EXIT, 1 );
  airportMenu.addCommand( exitCommand );
  airportMenu.setCommandListener( this );
  serviziMenu.addCommand( backCommand );
  serviziMenu.setCommandListener( this );
  display = Display.getDisplay( this );
 }
 public void startApp()
 {
   display.setCurrent( airportMenu );
 }
```

2 DEFINIAMO L'INTERFACCIA
Con questo codice creiamo una lista di aeroporti con i relativi codici, un menu con le opzioni di scelta e uno che mostra la lista precedentemente creata. Il primo menu che vedremo al lancio dell'applicazione sarà quello definito in *airportMenu*

```
public void commandAction( Command com,
                                Displayable dis )
{
   if ( dis == airportMenu && com ==
                         List.SELECT_COMMAND )
  {
    currentAirport = airportMenu.getSelectedIndex();
    serviziMenu.setTitle( aeroporti[
               currentAirport ] + " weather" );
    display.setCurrent( serviziMenu );
  }
  else if ( dis == serviziMenu && com ==
                         List.SELECT_COMMAND )
 {
   int choice = serviziMenu.getSelectedIndex();
   String risultato = "Some Error";
```

```
  switch( choice ) {
    case 0:
      SoapObject objResult = (SoapObject)
       callService(currentAirport, "getSummary" );
      if ( objResult != null ) {
         risultato = "Il tempo a " +
                           objResult.getProperty(0) +
                " è " + objResult.getProperty(3) +
        " con un cielo" + objResult.getProperty(2) +
" e il vento risulta " + objResult.getProperty(1) +
". L'umidità è " + objResult.getProperty(4) +
  ", la pressione è " + objResult.getProperty(5) +
 ", e la visibilità è " + objResult.getProperty(6) + ".";
        }
    break;
    case 1:
     risultato = (String) callService( (4)
                   currentAirport, "getHumidity" );
    break;
    case 2:
     risultato = (String) callService(
                   currentAirport, "getPressure" );
    break;
   }
   response.setString( risultato );
   display.setCurrent( response );
} else if ( com == backCommand ) {
  display.setCurrent( airportMenu );
} else if ( com == exitCommand ) {
  destroyApp( true );
  notifyDestroyed();
}
```

3 GESTIAMO I COMANDI
Con questa procedura gestiamo i comandi che l'utente impartirà dal cellulare. Notare che prende in input *dis* e *com*, rispettivamente contenenti il menu che si stava visualizzando e il comando impartito su quel menu. Se siamo nel menu dei servizi e viene impartito un comando *select*, si avvia una procedura di *case* per prelevare l'indice del comando e chiamare il Web Service con una procedura di callservice appropriata. Se siamo nel menu degli aeroporti viene selezionato l'aeroporto appropriato e il display viene settato a *servizimenu*.

```
private Object callService( int choice, String
                                    methodName )
{
  Object  risultato = null;
  try  {
     transport = new HttpTransport( serviceUrl,
              soapAction + "#" + methodName );
     transport.debug = true;
     classMap = new ClassMap();
```

```
        classMap.prefixMap = new PrefixMap(
    classMap.prefixMap, "air", serviceNamespace );
        transport.setClassMap( classMap );
        request = new SoapObject(
                serviceNamespace, methodName );
        request.addProperty( "arg0", codiciAeroporti[
                                            choice ] );
        risultato = transport.call( request );
    } catch( Exception e ) {
        e.printStackTrace();
        System.out.println( "Richiesta CALL: \n" +
                            transport.requestDump );
        System.out.println( "Risposta RESPONSE:
                  \n" + transport.responseDump );
        risultato = null;
    }
    return risultato;
}
```

4 RICHIAMIAMO IL WEB SERVICE
Questa è la procedura che realmente interroga il Web Service grazie a soap e restituisce il risultato appropriato.

Facile no? Ora quello che vi serve per capire quello che abbiamo fatto è il file *Meteo_Cellulare.zip* presente nel CD allegato alla rivista e che contiene l'esempio completo dell'applicazione.

RISULTATO FINALE

Passiamo al vero e proprio test dell'applicazione. Prima di tutto possiamo visualizzare cosa fa la nostra Midlet direttamente all'interno del Toolkit cliccando su *Run*. In questo modo, vediamo come si comporterebbe l'applicazione su un dispositivo standard (MIDP 1.0 o 2.0, a noi la scelta). In molti casi però l'esperienza insegna che una cosa è testare l'applicazione su un emulatore, un'altra è fare il deployment su un vero e proprio dispositivo. Quindi, dopo aver visto la nostra applicazione sul Toolkit andiamo nella cartella *bin* del nostro progetto e troviamo all'interno due file, *MeteoWS-.jad* e *MeteoWS.jar*. Questi due file servono per poter installare il programma su nostro cellulare. Per poter fare ciò bisogna avere un cavo di collegamento con il PC oppure avere un cellulare con dei collegamenti wireless come IRDA o Bluetooth. Una volta installato, andiamo ad aprire il programma e a richiedere delle informazioni. Ci apparirà una richiesta di connessione per poter accedere alla rete GPRS (le Midlet vengono eseguite, come le applet, in una sandbox per motivi di sicurezza). Subito dopo l'instaurazione della connessione, vedremo le informazioni richieste al WebService. In Rete possiamo trovare tantissimi altri WebService da utilizzare all'interno delle nostre applicazioni. Curiosando sul sito di riferimento per i WebService, *www.xmethods.net*, possiamo avere diversi spunti per nuove applicazioni. Stante la vastità dei servizi Web disponibili, solo la fantasia può porre un limite alle applicazioni realizzabili.

Federico Paparoni

Fig. 5: ***Installazione ed esecuzione della Midlet sul cellulare***

NOTA

INSTALLARE E PROVARE L'APPLICAZIONE

1) **Installare un JDK**
2) **Scaricare e installare il J2ME wireless toolkit 2.2 da** http://java.sun.com/products/j2mewtoolkit/download-2_2.html
3) **Scaricare le ksoap da** http://ksoap.objectweb.org/ **Il file da scaricare è ksoap-midp.zip**
4) **Dal menu start di windows/cartella J2ME Wireless toolkit 2.2/ lanciare la ksoap bar e creare un nuovo progetto dal nome MeteoWS. Il J2ME creerà una directory MeteoWS all'interno di Apps nella propria radice.**
5) **Rinominare il file ksoap-midb.zip in ksoap-midp.jar e copiarlo in MeteoWS/Lib**
6) **Copiare il file MeteoWS.java dal nostro cd nella directory MeteoWS/Src**
7) **Cliccare su "Run" nella ksoap bar**

Se tutto è andato a buon fine vedrete l'emulatore del telefonino con dentro l'applicazione appena creata.

Fig. 6: ***La risposta che viene data dal Web Service utilizzato dalla nostra Midlet***

NOTA

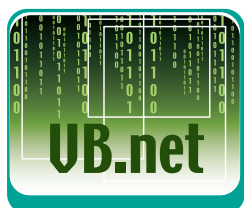
Una delle cose più importanti nell'utilizzo delle librerie kSOAP è il fatto che esistano diverse implementazioni di SOAP con diversi namespace. kSOAP utilizza il namespace *XML Schema 2001* mentre, ad esempio, l'implementazione SOAP di Apache è rimasta legata a XML Schema 1999. Esiste, comunque, il modo di utilizzare i servizi che adottano l'implementazione Apache, aggiungendo le seguenti righe di codice:

```
HttpTransport = new HttpTransport (serviceUrl, soapAction);
ClassMap classMap = new ClassMap (true);
httpTransport.setClassMap (classMap);
```

Istanziando in questa maniera la classe *ClassMap*, che si occupa di mappare le nostre richieste in XML, decidiamo di utilizzare la vecchia implementazione.

Al via il corso per chi vuole imparare a programmare

VB .NET 2003 Le basi

In questo articolo diamo uno sguardo al nuovo ambiente, impariamo come iniziare un progetto, creiamo un bottone e visualizziamo un messaggio

Utilizza questo spazio per le tue annotazioni

Visual Basic .NET 2003 è un linguaggio progettato per la piattaforma .NET di Microsoft, completamente orientato agli oggetti, supporta tutte le caratteristiche tipiche di un linguaggio Object-Oriented che, ad esempio, non erano presenti in Visual Basic 6. In questo primo appuntamento illustreremo l'ambiente di sviluppo di VB.NET e realizzeremo il nostro primo programma.

LA PAGINA INIZIALE

Avviando per la prima volta Visual Studio .NET 2003, dopo una piccola schermata di presentazione, viene mostrato l'ambiente di sviluppo integrato condiviso da Visual C#, Visual J#, Visual Basic e Visual C++. La pagina iniziale è suddivisa in tre schede: **Profilo Personale** ❸, **Progetti** ❶, **Risorse in linea** ❷.

La scheda Profilo Personale permette di impostare le preferenze per il comportamento dell'ambiente di sviluppo, adattandolo alle proprie esigenze. Analizziamo le possibili impostazioni:

- Dalla prima casella di riepilogo, alla voce **Profilo** ❹, è possibile selezionare uno dei profili predefiniti che impostano automaticamente: lo schema della tastiera, il layout finestra ed il filtro della *Guida* in base alle preferenze degli sviluppatori dei linguaggi disponibili. Nel nostro caso possiamo selezionare la voce *Sviluppatore Visual Basic*.
- Le opzioni in **Schema tastiera** ❺ consentono di impostare la mappatura della tastiera e le possibili configurazioni dei tasti di scelta rapida.
- Le opzioni **Layout finestra** ❻ consentono di selezionare le possibili configurazioni delle finestre nell'ambiente di sviluppo.
- La casella di riepilogo **Filtro Guida** ❼ permette di selezionare uno dei possibili filtri predefiniti per la documentazione contenuta in *Microsoft Developer Network* (MSDN). Selezionando l'opzione *Visual Basic,* ogni volta che si effettua la ricerca di un argomento nella guida, non saranno visualizzati elementi relativi agli altri linguaggi.
- I pulsanti di opzione **Visualizza Guida** ❽ consentono di selezionare le modalità di visualizzazione degli argomenti della guida in linea. L'opzione *Guida Interna* mostra la guida all'interno della stessa finestra in cui è visualizzata la schermata iniziale. L'opzione *Guida Esterna* consente di visualizzare gli argomenti in una finestra separata.
- L'elenco a discesa **All'avvio** ❾ consente di definire l'interfaccia utente che viene visualizzata al primo avvio di Visual Studio .NET. Le opzioni sono le seguenti: mostra pagina iniziale, carica ultima soluzione caricata, mostra finestra *Apri progetto,* mostra finestra nuovo progetto, visualizza ambiente vuoto.

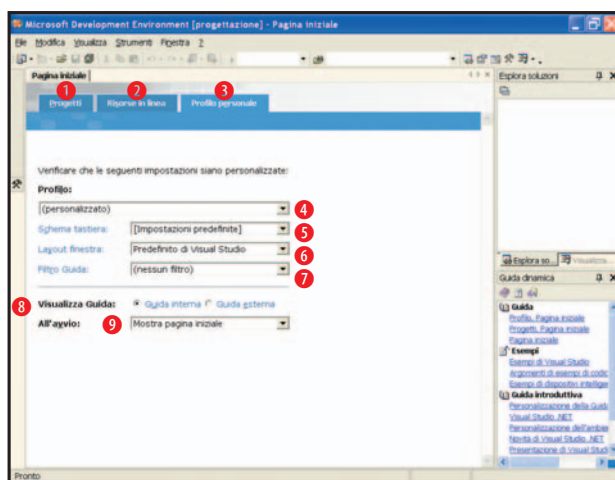



***Fig. 1:** La pagina iniziale dell'ambiente di sviluppo*

- La scheda **Risorse in linea** ❷ contiene collegamenti ad aree del sito Microsoft destinate agli sviluppatori.
- La scheda **Progetti** ❶ permette di accedere ai progetti esistenti facendo doppio click sulla lista dei progetti o selezionare un qualsiasi progetto cliccando sul pulsante *Apri Progetto*, o ancora di crearne di nuovi cliccando sul pulsante *Nuovo Progetto*. Per analizzare completamente l'ambiente di sviluppo è necessario creare un nuovo progetto.

LA BARRA DEI MENU ❿

La barra dei menu di Visual Studio .NET 2003 si autoconfigura, in base alle azioni effettuate. Nella finestra iniziale (prima della creazione di *PrimoProgramma*) la barra dei menù era costituita solo dai menu *File, Modifica, Visualizza, Strumenti, Finestra* e *Guida (?)*. Quando si crea una nuova applicazione si avrà di fronte la barra dei menu completa costituita dalle voci:

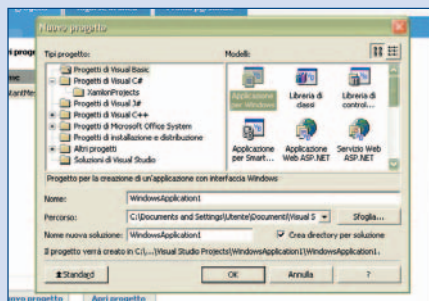
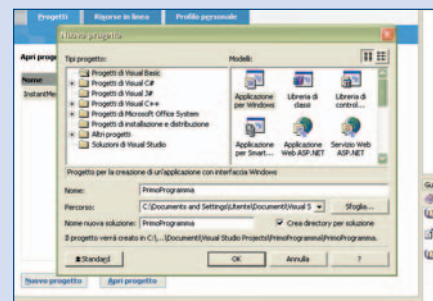
- **File, Modifica, Visualizza, Finestra:** sono i classici menu di Windows
- **Progetto**: permette di aggiungere vari tipi di file e riferimenti all'applicazione
- **Genera:** contiene le opzioni per trasformare il progetto in un'applicazione eseguibile
- **Debug:** fornisce gli strumenti per rintracciare ed eliminare gli errori (bug) all'interno di un'applicazione
- **Dati:** consente di utilizzare le informazioni provenienti da un database
- **Formato:** permette di manipolare l'aspetto dei controlli nelle finestre
- **Strumenti:** permette di configurare l'ambiente di sviluppo e di installare strumenti esterni
- **(?) Guida:** consente l'accesso alla documentazione di Visual Studio .NET 2003

LA BARRA DEGLI STRUMENTI ⓫

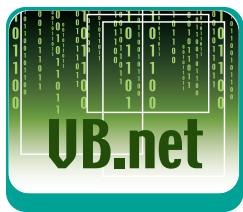
All'interno dell'ambiente di sviluppo sono disponibili molte barre degli strumenti che possono essere rimosse o aggiunte mediante le opzioni del menu: *Visualizza/Barre degli strumenti*. Ciascuna barra degli strumenti fornisce un accesso immediato ai comandi utilizzati con maggiore frequenza, evitando di esplorare una serie di sottomenu.
Ad esempio la prima icona (iniziando da sinistra) rappresenta il pulsante *Nuovo Progetto*, e cioè il comando disponibile nel menu *File/Nuovo/Progetto*.

Se dopo aver disegnato un controllo vi rendete conto che non si trova nella posizione desiderata oppure è troppo grande o troppo piccolo non preoccupatevi, è sempre possibile spostare o ridimensionare il controllo. Per spostare un controllo è sufficiente cliccare sul controllo e trascinarlo nella posizione desiderata. Per ridimensionare un controllo si deve trascinare uno dei quadratini bianchi ed allungare il controllo nella direzione desiderata.

COME CREARE UN NUOVO PROGETTO

1 **Cliccare sul pulsante *Nuovo progetto* della barra degli strumenti oppure selezionare la voce di menu *File/Nuovo/Progetto,* oppure premere contemporaneamente *Ctrl+Shift+N*. In alternativa utilizzare il pulsante *Nuovo Progetto.***

2 **Selezionare l'opzione *Progetti di Visual Basic* nella struttura ad albero della casella *Tipi Progetto,* posta a sinistra della finestra di dialogo. Selezionare l'opzione *Applicazione per Windows* nella casella *modelli* posta a destra.**

3 **Digitare il nome del progetto, nel nostro caso *PrimoProgramma,* nella casella di testo *Nome*. Se necessario salvare il progetto in un percorso diverso da quello predefinito premendo il tasto *Sfoglia*. Infine cliccare su *OK* per creare il primo progetto.**

VB.NET crea, quindi, una soluzione dal nome *PrimoProgramma,* costituita da un progetto *PrimoProgramma,* composto a sua volta da una finestra (form) vuota denominata *Form1.VB.*
A differenza di Visual Basic 6 non si parla più soltanto di progetti, ma di soluzioni.

Una soluzione può contenere diversi progetti scritti con linguaggi diversi e diverse tipologie di file. Se una qualsiasi delle finestre dell'ambiente di sviluppo, che analizzeremo in seguito, non è visibile si può utilizzare il menu *Visualizza* per selezionarle e visualizzarle. Se, inoltre, non vi soddisfa la collocazione di una particolare finestra, è possibile spostarla facilmente facendo clic sulla barra del titolo (la barra di colore blu nella parte superiore) e trascinandola nella nuova posizione. Le finestre dell'ambiente di sviluppo possono essere mobili, cioè indipendenti, oppure ancorate.

La barra degli strumenti che viene visualizzata all'apertura di Visual Studio è la barra denominata *Standard*.

LA FINESTRA CASELLA DEGLI STRUMENTI ⑫

La *Casella degli strumenti* contiene un elenco di controlli che è possibile disegnare sulle form in fase di progettazione. Per visualizzarla si può:

- Selezionare la voce di menu *Visualizza/Casella degli strumenti*
- Premere l'icona *Casella degli strumenti* sulla barra degli strumenti standard
- Premere i tasti *Ctrl+Alt+X*.

I vari controlli e componenti che possono essere collocati nel modulo sono divisi in schede all'interno della *Casella degli strumenti*.
Le freccie rivolte verso l'alto/basso accanto al titolo della scheda *Windows Forms* consentono di scorrere l'elenco di controlli *Windows Forms* verso l'alto/basso. Sono disponibili diversi set di controlli, in base al tipo di progettazione attiva nell'editor.

NOTA

Se nella scheda *Profilo* modifichiamo una delle impostazioni predefinite, dopo avere selezionato un profilo definito in precedenza, il nome del profilo viene modificato in "*(Personalizzato)*".

LA FINESTRA DEL CODICE ⑬

Nella finestra di disegno si verificano la maggior parte delle operazioni. In questa finestra viene disegnata l'interfaccia utente e scritto il codice. La finestra è strutturata in schede, navigando tra queste è possibile visualizzare, rapidamente, le finestre in cui disegnare l'interfaccia e le finestre per scrivere il codice. La finestra del codice possiede tutte le funzionalità dell'editor di VB6, tra cui l'evidenziazione sintattica tramite colori, l'IntelliSense e l'editing tramite drag-and-drop e ne aggiunge altre come ad esempio:

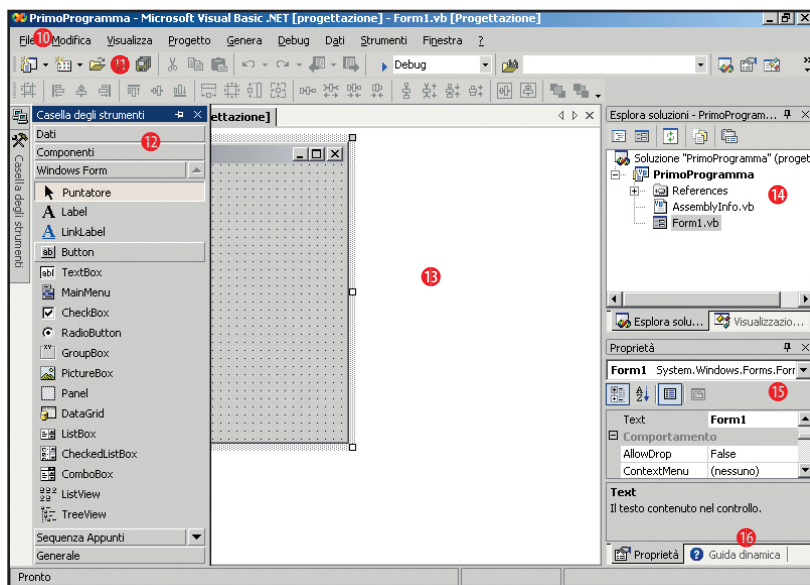
- **Rientri intelligenti:** l'editor di codice imposta automaticamente l'indentazione più logica per blocchi di codice come i cicli *For*, le istruzioni *If.. Then..Else*, ecc.
- **Selezione a blocchi:** è possibile selezionare blocchi rettangolari di codice tenendo premuto il tasto *Alt* mentre si trascina il mouse o si preme un tasto freccia
- **A capo automatico:** è possibile attivare questa funzionalità per mandare a capo, automaticamente, le righe di codice molto lunghe. (Per attivare o disattivare questa funzionalità è sufficiente selezionare la voce di menu *Modifica/Avanzate/A Capo Automatico.*

LA FINESTRA ESPLORA SOLUZIONI ⑭

La finestra *Esplora Soluzioni* contiene una visualizzazione delle soluzioni strutturata ad albero. Una soluzione rappresenta un gruppo di progetti (anche di linguaggio diverso) che "collaborano" tra loro per creare un'applicazione. La finestra fornisce informazioni in tempo reale sugli elementi della soluzione aperta e ne consente la gestione. Ad esempio, si possono spostare elementi da un progetto all'altro con semplici operazioni di drag-and-drop.
Rispetto alla precedente versione di VB.NET, la finestra *Esplora soluzioni* include tre nuove icone che evidenziano lo stato dei file condivisi in un team di sviluppo:

- **Archiviato:** il file è stato archiviato in un database di controllo del codice sorgente.
- **Estratto in esclusiva:** il file viene estratto, dal database di controllo, da un solo sviluppatore. Gli altri sviluppatori non possono accedere al file.
- **Checked Out Shared:** il file viene estratto, dal database di controllo, per essere condiviso dal team di sviluppo ed al momento del rilascio le diverse versioni vengono unificate.

LA FINESTRA PROPRIETÀ ⑮

La *Finestra Proprietà* mostra le proprietà dell'oggetto selezionato. Ogni componente gode di alcune proprietà che possono essere visualizzate in ordine alfabetico oppure divise per categorie, modificabili tramite questa finestra. Per alcune proprietà, posizionandoci con il mouse sul campo alla propria

destra, è disponibile un elenco di impostazioni predefinite che è possibile visualizzare cliccando sulla freccia rivolta verso il basso. Facendo doppio clic sul nome di una proprietà è possibile scorrere tutti i valori che la proprietà può assumere. Le proprietà possono anche essere impostate da codice.
Per attivare velocemente la finestra si può premere il tasto *F4*.

LA FINESTRA GUIDA DINAMICA 16

La finestra *Guida dinamica* consente, con un semplice clic, di accedere ad argomenti contestuali, relativi cioè, a ciò che è attivo nell'ambiente di sviluppo.

IL NOSTRO PRIMO PROGRAMMA

Dopo aver seguito i passi nel box *"Come creare un nuovo progetto".* La prima operazione sarà quella di ridenominare il file fisico che contiene *Form1.VB* in un nome più significativo, per farlo si deve:

- Cliccare su *Form1.vb* nella finestra *Esplora Soluzioni.*
- Da *Finestra Proprietà*, si può modificare la proprietà *Nome File* da *form1.vb* a *PrimaForm.vb* e premere invio

Si può notare come il nome della finestra venga aggiornato nella finestra *Esplora Soluzioni.* Cliccando sulla *Form,* visualizzata nella finestra di disegno, la finestra delle proprietà sarà notevolmente diversa, poiché mostrerà le proprietà dell'oggetto *Form* anziché le proprietà del file che lo contiene (viste in precedenza). Prestate attenzione, la differenza è il risultato di due visualizzazioni diverse dello stesso file. Quando il nome della form è evidenziato nella finestra *Esplora Soluzioni,* vengono visualizzate le proprietà fisiche. Quando la form è evidenziata nella finestra di *Disegno,* vengono visualizzate le proprietà visive e quelle logiche. Le proprietà sono raggruppate (per default) in categorie, ad esempio nella categoria *Aspetto* si trovano proprietà come *BackColor,* che serve ad impostare i colori di sfondo della form, *Font* per impostare il carattere utilizzato per il testo, e *Text* che permette di impostare l'intestazione visualizzata sulla barra del titolo. Per visualizzare le proprietà in ordine alfabetico si deve cliccare sul piccolo pulsante *AZ* presente nella parte superiore della finestra. Possiamo modificare il titolo della *Form* modificando il valore della proprietà *Text,* nella finestra delle proprietà, in: *PrimaForm.* La successiva operazione sarà di disegnare un controllo *Button* sulla *Form.* Nel momento in cui l'utente clicca su questo pulsante, verrà visualizzato un messaggio di benvenuto.
Disegnare un controllo su una form è un'operazione semplicissima, analizziamo in dettaglio le operazioni necessarie:

- Cliccare sull'icona *Button* presente nella barra degli strumenti.
- Spostare il puntatore sulla form. Noterete che il puntatore assume la forma del mirino con accanto l'icona del componente selezionato.
- Tenere premuto il pulsante del mouse nel punto in cui si desidera collocare l'angolo superiore sinistro dell'etichetta e trascinare il puntatore nel punto in cui di desidera collocare l'angolo inferiore destro.

Il posizionamento dei controlli sulla form può anche essere effettuato facendo doppio clic sul controllo nella barra degli strumenti, in questo caso i controlli vengono disegnati nell'angolo superiore sinistro della form. Dopo aver disegnato il bottone sulla finestra, le prime due operazioni saranno quelle di modificarne il nome ed il testo visualizzato, per farlo, selezioniamo il bottone e nella finestra delle proprietà cambiamo la proprietà *Name* in *ButtonMessaggio* e la proprietà *Text* in *Visualizza Messaggio*. A questo punto si può passare a scrivere il codice. Per visualizzare la finestra del codice è sufficiente fare doppio clic sul bottone, si aprirà una nuova finestra con il cursore posizionato all'interno della procedura di evento *ButtonMessaggio_Click*.

Fig. 2: **Il programma in esecuzione**

Il codice contenuto all'interno della procedura *ButtonMessaggio_Click* verrà eseguito ogni volta che l'utente clicca con il mouse sul pulsante, per questo al suo interno scriviamo:

```
Private Sub ButtonMessaggio_Click(ByVal sender
                    As System.Object, ByVal e As
 System.EventArgs) Handles ButtonMessaggio.Click
   MessageBox.Show("Benvenuti nel mio Primo Programma")
End Sub
```

La classe *MessageBox* espone il metodo *Show* che permette di visualizzare una finestra di messaggio che può contenere testo, pulsanti e simboli che forniscono all'utente istruzioni e informazioni. Il programma è ora terminato. Fate clic sul pulsante *Avvia* della barra degli strumenti. Verrà visualizzata una finestra con un bel pulsante al centro, e cliccando su di esso verrà mostrato il messaggio di benvenuto.

Luigi Buono

LA FINESTRA ESPLORA SERVER

La finestra *Esplora server* fornisce l'accesso per lo sviluppo lato server. Consiste in una finestra condivisa, che aiuta gli sviluppatori ad accedere e a gestire le risorse di qualsiasi computer (locale o remoto), per cui si possiedono i permessi appropriati. Con l'Esplora server, sarà possibile connettersi ai server e visualizzarne le risorse.

Creare le interfacce con "Standard Widget Toolkit"

Java, sviluppare con Eclipse 3

In questo numero realizziamo finestre verticali e orizzontali, gestiamo i margini e impariamo a posizionare i bottoni

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste

Programmazione Java

Software

Eclipse 3.0.1 e JDK 1.4.2 o superiore su qualunque sistema operativo supportato (Linux, HP-UX, Solaris, AIX, Windows, MacOS)

Impegno

Tempo di realizzazione

Standard Widget Toolkit è una delle più potenti librerie grafiche per la creazione di interfacce in Java. Ne introduciamo gli aspetti di base. Lo scopo è capire quali opportunità ci vengono offerte per disporre controlli quali ad esempio bottoni o finestre di testo all'interno di un'applicazione. Tornando ai layout di SWT, invece, avevamo già dato un breve elenco di quali siano i layout disponibili nel framework (si tenga presente che se ne possono creare di ulteriori personalizzati). Vediamoli in maniera meno sintetica, dando qualche accenno di utilizzo pratico.

IL METODO PIÙ SEMPLICE

La cosa più facile che si può fare con SWT è creare una finestra i cui controlli siano disposti orizzontalmente o verticalmente a uguale distanza fra di loro. Per uniformarci alla nomenclatura tradizionale ci riferiremo all'oggetto che ci consente di gestire i vari posizionamenti, con il nome di "layout manager". Come già detto il layout manager più semplice da usare è quello che pone i controlli a uguale distanza fra loro. Tale layout manager è denominato *FillLayout*.

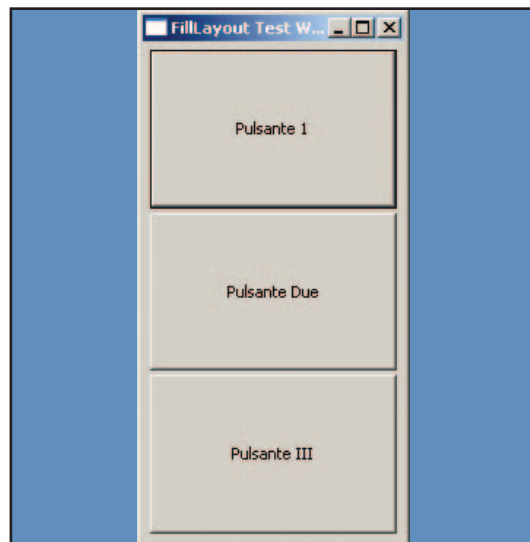



***Fig. 1:** Una finestra con tre pulsanti posizionati con un layout di tipo "fillLayout" verticale.*

Il costruttore della classe ha un parametro solo, lo stile. Può assumere due valori: *HORIZONTAL*, e *VERTICAL*, a seconda che i bottoni debbano essere disposti in orizzontale o in verticale nella finestra. Le uniche opzioni disponibili sono: il numero di pixel da lasciare come margine dai bordi, rispettivamente *marginWidth* e *marginHeight*, e la distanza, sempre in pixel, tra i vari elementi ovvero *spacing*.

Di seguito troverete il codice che genera la finestra della **Figura 1**: notate come sia possibile anche creare un *FillLayout* senza parametri di costruzione ed utilizzare poi la proprietà *type* per indicare lo stile.

```
private void createSShell() {
  sShell = new Shell();
  FillLayout fillLayout1 = new FillLayout();
  fillLayout1.type = org.eclipse.swt.SWT.VERTICAL;
  fillLayout1.marginHeight = 5;
  fillLayout1.marginWidth = 5;
  fillLayout1.spacing = 3;
  sShell.setLayout(fillLayout1);
  sShell.setSize(new org.eclipse.swt.graphics.Point(
                                                187,369));
  sShell.setText("FillLayout Test Window");
  button = new Button(sShell, SWT.NONE);
  button2 = new Button(sShell, SWT.NONE);
  button1 = new Button(sShell, SWT.NONE);
         button.setText("Pulsante 1");
         button2.setText("Pulsante Due");
         button1.setText("Pulsante III");
}
```

UN PO' DI FLESSIBILITÀ

Se *fillLayout* era semplice, *RowLayout* non è complesso. A differenza del primo, possiamo definire margini separati per sopra, sotto, destra e sinistra (rispettivamente *marginTop, marginBottom, marginRight, marginLeft*). Abbiamo poi la possibilità di dare ad ogni widget inserito nella riga o colonna una certa dimensione. È sufficiente associare al widget tramite *setLayoutData* un oggetto di tipo *RowData*, il quale espone due proprietà, *width* e *height*, per impostare rispettivamente larghezza ed altezza.

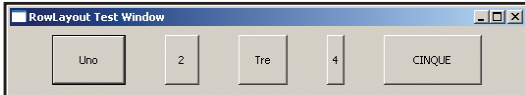



Fig. 2: ***Un esempio di Rowdate con justify, pack e fill vere. Da notare che gli elementi sono equidistanti, hanno dimensioni diverse e la loro altezza è identica***

La **Figura 2** è un esempio di *RowLayout*. Tra le opzioni della classe abbiamo *justify*, che se *true* fa sì che lo spazio che avanza nel container venga ridistribuito tra le spaziature dei vari elementi. *Pack*, che quando *false* rende tutti gli elementi della stessa dimensione in modo da essere uguali all'elemento maggiore. *Fill*, il quale, messo a *true*, fa sì che l'altezza, per le righe, o la larghezza, per le colonne, di tutti i widget sia pari al valore più alto tra quelli impostati con *RowData*. Infine *wrap*, che se vale *true* fa sì che, a seconda delle esigenze di spazio, vengano create più di una sola riga o colonna.

Fig. 3: ***Un esempio di RowLayout con justify e fill false***

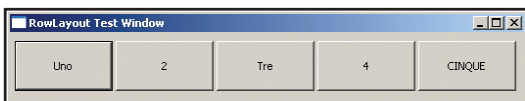



Fig. 4: ***Un esempio di RowLayout con pack settato a false. Da notare che i pulsanti hanno tutti la stessa dimensione***

UN LAYOUT STRANO

Stiamo parlando di *StackLayout*: con esso tutti gli elementi aggiunti avranno la stessa posizione e dimensione. Di conseguenza, uno solo, quello puntato dalla proprietà *topControl*, sarà effettivamente visibile. A run-time è possibile modificare l'elemento visualizzato cambiando il valore di *topControl* ed invocando il metodo *layout* del contenitore per ridisegnare la finestra. Le uniche opzioni con questa classe di gestione del posizionamento sono i margini laterali e verticali, *marginWidth* e *marginHeight*, espressi in numero di pixel, esattamente come in *RowLayout* e *FillLayout*.

IL PIÙ POTENTE

Il gestore in questione è *FormLayout*: si tratta di una classe nuova appena introdotta nella versione 2.0 di SWT. Di per sé la classe offre solo tre proprietà, che abbiamo tra l'altro visto precedentemente più volte e che sono *marginWidth, marginHeight* e *spacing*. Ma il concetto fondamentale, quello che dà tanta flessibilità a questo strumento, è che ad ognuno dei widget è possibile associare un *FormData*, tramite il quale si possono definire svariate modalità con cui un lato del widget stesso si rapporta al container o ad altri elementi. Il *FormData* di base può essere usato per definire la dimensione del componente a cui viene legato (proprietà *width* e *height*): espone, però, anche quattro proprietà per ognuno dei suoi lati (*top, bottom, right, left*) che possono contenere istanze di FormAttachment.
Ecco quindi, pian piano, i passi per la creazione di una finestra che utilizza il nostro nuovo layout. Per prima cosa utilizziamo il metodo *setLayout* per impostare il layout del container su un'istanza di *FormLayout* configurata con i margini e la spaziatura che desideriamo:

```
Display display = new Display ();
   Shell shell = new Shell(display);
   FormLayout layout = new FormLayout();
   layout.marginWidth = 5;
   layout.marginHeight = 5;
   layout.spacing = 3;
   shell.setLayout(layout);
```

Dopodiché possiamo cominciare a creare i vari widget, insieme ai *FormData* e agli eventuali attachment da associare loro per posizionarli come desideriamo:

```
Button button = new Button(shell, SWT.PUSH);
   button1.setText("BUTTON");
   FormData data = new FormData(50,50);
   data.top = new FormAttachment(50, - data.height / 2);
   data.left = new FormAttachment(50, - data.width / 2);
   button.setLayoutData(data);
```

Ci sono tre possibilità per passare dei parametri alle proprietà *top, left* e *right* di un attachment. Nel caso più semplice abbiamo un solo parametro di tipo intero, con cui indichiamo a quale percentuale della dimensione della finestra vogliamo che sia posizionato il lato in questione. Se il lato è *right* o *left*, la percentuale indicata è relati-

NOTA

CHE COSA È UN LAYOUT

Bisogna immaginare una finestra come divisa in blocchi rettangolari di diversa grandezza. I controlli come bottoni, label etc. possono essere posizionati in questi spazi rettangolari. Un layout contiene metodi per disporre gli oggetti nei blocchi così come esso li espone.

CHE COSA È UNA SHELL

È possibile creare dei layout personalizzati derivando dalla classe *org.eclipse.swt.widgets.Layout*, implementando poi i metodi *computeSize* per restituire la dimensione che un determinato componente dovrà avere in base ai suoi eventuali vincoli di posizionamento, e layout con cui si posizionano fisicamente i vari elementi all'interno dell'area client del loro contenitore.

CHE COSA È UN DISPLAY

Il display rappresenta una sezione di lavoro di SWT.
È il responsabile della comunicazione con il sistema operativo.

va alla larghezza della finestra. Per *top* e *bottom* si prende in considerazione l'altezza. Nel secondo caso passiamo due int (come nell'esempio) che sono sempre la percentuale e in più un offset di distanziamento da tale percentuale in pixel. Dovreste ora aver intuito che nel codice mostrato noi centravamo il pulsante esattamente a metà del form (50% dell'altezza del form meno la metà dell'altezza del componente, e stessa cosa per la larghezza) il risultato lo vedete in **Figura 5**. La terza possibilità prevede l'utilizzo di numeratore e denominatore per determinare il punto di posizionamento relativo al container, più il solito offset in pixel.

Fig. 5: ***Il pulsante posizionato esattamente al centro della finestra grazie all'uso di una percentuale e un offset***

Ecco quindi come si potrebbe centrare un pulsante esattamente ai 2/3 del container:

```
data.top = new FormAttachment(2, 3, - data.height / 2);
data.left = new FormAttachment(2, 3, - data.width / 2);
```

Il risultato lo vedete in **Figura 6**.

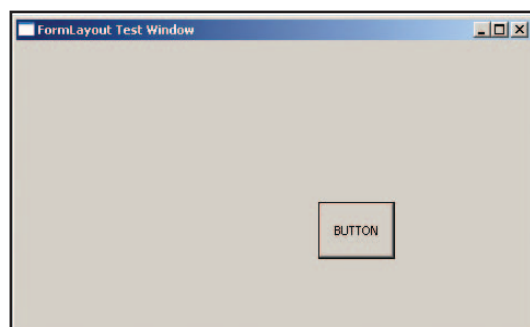



Fig. 6: ***Il pulsante posizionato a 2/3 del container grazie all'uso di numeratore e denominatore più un offset***

ATTACHMENT CIRCOLARI

Attenzione a non definire degli attachment circolari, per esempio un pulsante il cui lato destro è allineato col sinistro di un altro pulsante, il cui lato sinistro a sua volta è legato al destro del primo pulsante. In questo caso, SWT riesce ad interrompere il ciclo, senza entrare quindi in un loop infinito, ma il risultato visuale sarà indefinito.

Eseguendo il codice di esempio allegato alla rivista, potete anche sperimentare come le proporzioni e i vincoli definiti dagli attachment vengano rispettati anche ridimensionando la finestra. In alternativa alle capacità viste fin'ora, *FormAttachment* permette, inoltre, di ancorare un oggetto ad un altro elemento dello stesso contenitore. Anche per questa modalità ci vengono offerti dall'API di SWT tre costruttori. Il primo richiede solo di specificare l'elemento da cui partire: il lato impostato sarà adiacente al controllo passato come parametro con, eventualmente, una distanza uguale allo spacing impostato nel layout. Provate ad immaginare cosa fa il seguente blocco, prima di dare un'occhiata alla **Figura 7**:

```
FormData data = new FormData(70,50);
data.top = new FormAttachment(1, 3, - data.height / 2);
data.left = new FormAttachment(1, 3, - data.width / 2);
button1.setLayoutData(data);
Button button2 = new Button(sShell, SWT.PUSH);
button2.setText("OTHERB");
data = new FormData(70,50);
data.top = new FormAttachment(button1);
data.left = new FormAttachment(button1);
button2.setLayoutData(data);
```

La seconda possibilità offerta da questa nuova serie di costruttori aggiunge semplicemente un offset in pixel per distanziare il widget dal componente scelto come riferimento.
La terza versione permette di specificare anche un intero che rappresenta una costante di allineamento rispetto all'elemento in questione. Tale allineamento può variare il punto di adiacenza di due elementi. Normalmente, la destra viene accostata alla sinistra e la sinistra alla destra. Ma se scegliamo un valore di allineamento diverso, possiamo far coincidere la sinistra del primo elemento con la sinistra del secondo in altezza.
Per fare coincidere quindi la posizione verticale dei due pulsanti dell'esempio precedente, era quindi possibile scrivere, con lo stesso risultato, ma di modo formalmente più corretto:

```
data.top = new FormAttachment(button1, 0, SWT.TOP);
```

APRIRE I PROGETTI ALLEGATI

In allegato all'articolo trovate tutto il workspace utilizzato per creare le applicazioni di esempio. Potete aprirlo estraendo il file .zip e selezionando la cartella workspace risultante come nuovo ambiente di lavoro tramite il menu *File->Switch Workspace*… Alternativamente potete importare i singoli progetti che si trovano sotto la cartella indicata (in questo caso *JavaDevWithEclipse* o *PdfGeneratorWithEclipse*) tramite il menu *Import*… con origine dati *Existing Project into Workspace*.

ANTICIPAZIONI

Questa volta abbiamo davvero gettato le basi per poter creare, seriamente, delle interfacce Standard Widget Toolkit per applicazioni di qualunque portata. Ciò che ci aspetta adesso quindi è la creazione del codice di gestione del processo di creazione di un file PDF. Nel prossimo numero introdurremo la libreria Java per la generazione di file PDF.

Federico Mestrone

Gestione avanzata delle eccezioni in Java

Errori imprevedibili: Gestiamoli!

In questo articolo impereremo qualcosa sugli errori, gestiremo quelli imprevedibili, diremo qualcosa sul design del software e ci difenderemo dalle nullpoint exception

Utilizza questo spazio per le tue annotazioni

Sai già che un programma Java lancia un'eccezione quando incontra un problema durante l'esecuzione del programma (si dice a tempo di esecuzione, o a runtime). La maggior parte di questi problemi è la conseguenza di un bug, un errore commesso dal programmatore durante la scrittura del programma (si dice a tempo di compilazione, o a *compile time*, o anche a *design time*). Ma non tutti i problemi che si verificano a runtime sono bug. Ne esistono alcuni che semplicemente non si possono prevedere né evitare a tempo di compilazione. Cosa succede se cerchi di scrivere un file su un disco rigido pieno? O se cerchi di connetterti ad un server non disponibile? O se accedi ad un database senza avere i necessari permessi? Ci sono molte operazioni "a rischio" come queste, soprattutto quelle che richiedono l'accesso a risorse esterne (il file system, la rete...). Di solito un programmatore Java distingue esplicitamente tra le eccezioni che derivano da veri e propri bug e quelle che derivano da problemi di sistema a runtime. Le prime si rappresentano con una eccezione *unchecked*, cioè una sottoclasse di *RuntimeException;* le seconde con un'eccezione checked, cioè una sottoclasse di *Exception* ma non di *RuntimeException*. Le eccezioni di entrambi i tipi si propagano attraverso il codice fino a quando non vengono gestite in un blocco *try-catch*, o fino a quando non escono dal *main()* terminando il programma. La differenza è che le eccezioni *unchecked* si propagano silenziosamente, mentre le eccezioni *checked* sono molto più "rumorose": ogni volta che un metodo lancia un'eccezione checked, lo deve dichiarare esplicitamente. Vediamo un esempio:

```
public class PortaNonDisponibile extends Exception {
   public PortaNonDisponibile(String msg) {
      super(msg); }
}
```

Di solito tutte le eccezioni hanno un nome che termina con *Exception*, ma in questo esempio ho voluto evitare nomi troppo lunghi. Nel sistema ipotetico che sto scrivendo, *PortaNonDisponibile* è un'eccezione che indica che la porta sulla quale vogliamo stabilire una connessione di rete è già occupata. È un'eccezione *checked*, perché deriva da *Exception* ma non da *RuntimeException*. Ecco una classe che rappresenta una connessione di rete:

```
public class Connessione {
   private static int NUMERO_PORTE = 2;
   private final int porta;
   public Connessione(int porta) {
      this.porta = porta; }
   public void apri() {  // errore!
      if(porta <= 0 || porta > NUMERO_PORTE)
         throw new PortaNonDisponibile("La porta "
                              + porta + " non e' disponibile");
      System.out.println("Apertura connessione
                                 su porta " + porta); }
}
```

Una *Connessione* è associata ad una porta, che viene passata al costruttore di Connessione sotto forma di int. Quando un client cerca di aprire la Connessione, quest'ultima controlla che la porta sia compresa in un certo intervallo (nel nostro caso avremmo potuto fare questo controllo anche nel costruttore, perché il valore deve sempre essere compreso tra 1 e la costante *NUMERO_PORTE* , ma immagina che l'intervallo delle porte disponibili possa cambiare dinamicamente a *runtime*). Se la porta è fuori da questo intervallo, la Connessione lancia una eccezione *PortaNonDisponibile*. Questo codice, però, non compila. Il compilatore si accorge che il metodo *apri()* può lanciare una *PortaNonDisponibile*, e pretende che questa eccezione venga esplicitamente dichiarata dal metodo. Un metodo dichiara di poter lanciare un'eccezione con la parola chiave *throws* (da non confondere con *throw*, che lancia materialmente

l'eccezione), che deve essere aggiunta subito prima della parentesi graffa:

```
public class Connessione...
    public void apri() throws PortaNonDisponibile {
        <...> }
```

Un metodo può anche dichiarare più eccezioni, se le separa con una virgola:

```
public void f() throws Eccezione1, Eccezione2,
                                    Eccezione3...
```

Questa regola vale per tutte le eccezioni, ma diventa obbligatoria quando si ha a che fare con eccezioni *checked*. Un metodo può dichiarare le proprie *RuntimeException*, ma non è costretto a farlo. L'idea è che le eccezioni *unchecked* potrebbero saltar fuori più o meno ovunque, quindi sarebbe un lavoraccio dichiararle tutte. Le eccezioni *checked*, al contrario, sono in qualche modo *"previste"*: segnalano problemi che potrebbero verificarsi a runtime, ma che speriamo non si verifichino – e che in ogni caso non abbiamo modo di evitare. Quando un metodo dichiara una di queste eccezioni, costringe il suo client a prevederla e gestirla in qualche modo.

ECCEZIONI LATO CLIENT

Come fa il client di un metodo a gestire un'eccezione? Ha due possibilità: se non sa cosa farne, può a sua volta rilanciarla al proprio client; se sa cosa farne, può afferrarla e gestirla con un blocco *try-catch*. Proviamo a scrivere un client della classe *Connessione:*

```
public class Comunicatore {
    public static void main(String[] args) {
        apriConnessioni();
    }
    public static void apriConnessioni() {
        for(int i = 0; i < 4; i++) {
            apriConnessione(i);
    }
    private static void apriConnessione(int porta) {
        Connessione c = new Connessione(porta);
        c.apri();  // errore!
    }
}
```

Abbiamo un altro errore di compilazione. Il metodo *apriConnessione()* chiama *Connessione.apri()*, che dichiara di poter lanciare una *PortaNonDisponibile*. Ma *apriConnessione()* non gestisce questa eccezione, né la dichiara a sua volta. Per risolvere il problema possiamo aggiungere la dichiarazione dell'eccezione anche a questo metodo:

```
public class Comunicatore...
    public static void apriConnessioni() {
        for(int i = 0; i < 4; i++) {
            apriConnessione(i);  // errore! }
    private static void apriConnessione(int porta)
                              throws PortaNonDisponibile {
        Connessione c = new Connessione(porta);
        c.apri();}
}
```

Ora l'eventuale eccezione di *PortaNonDisponibile* verrà automaticamente rilanciata da *apriConnessione()* al proprio chiamante, proprio come una *RuntimeException*. Ma abbiamo solo trasferito il fardello sulle spalle del metodo chiamante, *apriConnessioni()*. Possiamo continuare a dichiarare l'eccezione anche in questo metodo (e trasferire il problema sul povero *main()*), oppure possiamo gestire l'eccezione con un blocco *try-catch*. Ho scelto questa seconda strada:

```
public class Comunicatore...
    public static void apriConnessioni() {
        for(int i = 0; i < 4; i++) {
            try {
                apriConnessione(i);
            } catch (PortaNonDisponibile e) {
                System.out.println("Fallita apertura porta " + i);
                richiediPorta(i); }
        }
        System.out.println("Fine di apriConnessioni()");}
    private static void richiediPorta(int porta) {
        System.out.println("Richiesta apertura porta "
                                              + porta);
    }
}
```

Fig. 1: *Lo schema delle classi utili a gestire le eccezioni*

LA PIAGA DEI CATCH VUOTI

Quando lavorerai sul codice Java di qualcun altro, probabilmente ti capiterà spesso di incontrare situazioni come questa:

```
try {
  x.metodoCheDichiara
Eccezioni();
} catch (Exception e) {}
```

Molti programmatori usano questa scappatoia per buttare via le eccezioni quando non sanno o non vogliono gestirle. Questo è male, perché finisce per nascondere il fatto che qualcosa non è andato per il verso giusto a runtime.

NOTA

IL MONDO REALE

Nelle librerie Java, molte classi lanciano eccezioni *checked*. Un esempio: la classe *File* contiene anche metodi statici come *File.createTempFile()*, che crea un file temporaneo sul disco rigido dell'utente. Se il metodo non riesce a creare il file (ad esempio perché il disco rigido è pieno), allora lancia una *IOException* per segnalare che qualcosa è andato storto durante le operazioni di IO (*Input-Output*). Tutte le chiamate a questo metodo devono mettere in conto e gestire questa eccezione.

Ora la chiamata ad *apriConnessione()* risiede in un blocco *try-catch*. Se una *PortaNonDisponibile* dovesse saltar fuori da questa chiamata, il controllo passerebbe al blocco *catch*. Il *catch* prende le adeguate contromisure: segnala il problema all'utente (in realtà, rozzamente, stampa qualche informazione sullo schermo) e chiama il metodo *richiediPorta()*. Nel nostro esempio questo metodo non fa niente di utile, ma ancora una volta ti chiedo di usare la fantasia: in un sistema reale, *richiediPorta()* potrebbe segnalare al sistema che la porta in questione è stata richiesta e dovrebbe essere liberata per un eventuale tentativo successivo.
Ecco il risultato del programma e la risposta alla domanda: quale sarà l'output del programma Comunicatore?

```
Fallita apertura porta 0
Richiesta apertura porta 0
Apertura connessione su porta 1
Apertura connessione su porta 2
Fallita apertura porta 3
Richiesta apertura porta 3
Fine di apriConnessioni()
```

Avevi indovinato?
Riassumiamo la filosofia di base delle eccezioni:

- di solito il metodo dove si verifica un errore non è lo stesso che sa come gestire l'errore, quindi...

- ...l'errore viene passato su per la catena dei chiamanti, fino a quando non incontra qualcuno che sa come gestirlo.

Questo passaggio dell'eccezione attraverso una serie di potenziali gestori, fino a trovare uno che se ne assuma la responsabilità, è quello che si chiama una catena di responsabilità. Nel CD puoi trovare un altro esempio (che per brevità non trascrivo in questo articolo) di nome *CatenaDiResponsabilita.java*.

PROGRAMMARE IN DIFESA

Dopo tanto parlarne, amico lettore, potresti pensare che le eccezioni siano una specie di flagello dal quale stare continuamente in guardia. In effetti alcuni programmatori si sentono a disagio quando scrivono anche una sola riga di codice senza gestire le eccezioni che potrebbero derivarne. Ma sarebbe pressoché impossibile dichiarare o gestire tutte le eccezioni che potrebbero saltare fuori da un programma, ed è questo il motivo per cui esistono le *RuntimeException*.
Una delle eccezioni più comuni e invise ai programmatori Java è la *NullPointerException*. Il sistema ne lancia una se cerchi di accedere ad un oggetto che non è stato inizializzato. Questa eccezione può saltare fuori più o meno da qualsiasi metodo. Ecco un esempio:

```
public class Comunicatore2 {
  private static Connessione conn;
  public static void main(String[] args)
  {
     apriConnessione(conn);
  }
  public static void apriConnessione(Connessione c) {
     try {
        c.apri();
     } catch (PortaNonDisponibile e) {
        e.printStackTrace();
     }
  }
}
```

Ho usato il metodo *printStackTrace()* della classe *Exception*, che stampa sullo standard output tutte le informazioni importanti contenute nell'eccezione (è lo stesso metodo che viene chiamato dalla JVM quando l'eccezione salta fuori dal *main()*).
Il metodo *Comunicatore.apriConnessione()* riceve una Connessione e la apre. Un piccolo problema: mentre scrivevo questo semplice programma, ho dimenticato una riga di codice. Quella che inizializza il campo *c*. Avrei dovuto scrivere qualcosa di simile:

NOTA

LA FAMIGLIA DELLE ECCEZIONI

Le classi che vedi in Fig. 1 fanno parte delle librerie standard di Java. Vediamo cosa significano. Alla radice delle eccezioni c'è la classe Throwable. Questa classe identifica tutto ciò che può essere "lanciato" da un programma Java per segnalare che qualcosa non è andato per il verso giusto. Throwable ha due sottoclassi: Error e Exception. Di Error parleremo tra poco. Exception è la capostipite di tutte le eccezioni, come StringIndexOutOfBounds Exception.Exception ha molte sottoclassi, ma una in particolare è molto importante: si chiama RuntimeException, e se hai letto l'articolo del mese scorso la conosci sai. RuntimeException è la madre di tutte le eccezioni cosiddette unchecked. Tutte le eccezioni che derivano da Exception ma non da RuntimeException si chiamano invece eccezioni checked (che letteralmente significa "controllate"). Questo mese parleremo di questa seconda categoria di eccezioni. Quella che vedi nella figura non è che la radice della gerarchia delle eccezioni. Al di sotto di queste classe ce ne sono altre dozzine, ciascuna delle quali è un tipo specifico di eccezione. Se cerchi di accedere fuori dai limiti di una stringa, Java lancerà una StringIndexOutOfBoundsException; se cerchi di usare un oggetto prima di averlo creato otterrai una NullPointerException; e così via.

```
public class Comunicatore...
    private static final int PORTA_DI_DEFAULT = 1;
    private static Connessione conn = new
                        Connessione(PORTA_DI_DEFAULT);
```

Invece al reference non viene mai assegnato un oggetto, quindi il suo valore resta quello di default. Il valore di default per un reference è *null*, che indica che un reference non punta a niente. Se provi a far girare la prima versione di questo programma otterrai un messaggio di errore simile a questo:

```
java.lang.NullPointerException
   at it.ioprogrammo.corsojava.eccezioni.Comunicatore
                .apriConnessione(Comunicatore.java:12)
   at it.ioprogrammo.corsojava.eccezioni.Comunicatore
                          .main(Comunicatore.java:8)
Exception in thread "main"
```

Java lancia una *NullPointerException* ogni volta che cerchi di accedere ad un campo o di chiamare un metodo su un reference null (questa eccezione dovrebbe più correttamente chiamarsi *NullReferenceException*, ma per motivi storici dobbiamo tenerci il nome che ha). Nel nostro esempio è facile capire da dove viene l'errore; ma in un programma più complicato le *NullPointerException* possono essere frequenti, e può essere difficile trovarne le cause. Cosa succederebbe se qualche altra classe chiamasse il metodo *apriConnessione()* passandogli un oggetto creato magari da una terza classe, in un punto molto lontano del codice? Diventerebbe difficile garantire che tutte le chiamate a questo metodo (o a qualsiasi altro metodo) ricevano parametri inizializzati correttamente. Per proteggersi dalle *NullPointerException*, alcuni programmatori scelgono di "programmare in difesa", e infarciscono il codice di controlli come questo:

```
public class Comunicatore...
    public static void apriConnessione(Connessione c) {
        if(c == null)
            return;
        <...>
    }
```

Ti consiglio di non farti prendere la mano da questa strategia. La responsabilità di garantire che i parametri siano inizializzati bene non è in generale di chi li riceve, ma di chi li crea. Piuttosto che ricorrere a queste cure sintomatiche, meglio risolvere il problema alla radice. In particolare, se vuoi evitare le *NullPointerException* dovresti inizializzare tutti i riferimenti agli oggetti il prima possibile, possibilmente nella stessa riga in cui li dichiari. Ogni reference null che va in giro per il tuo programma è potenzialmente rischioso.

PROPRIO NIENTE DA FARE

Abbiamo detto che la classe *Exception* è figlia di *Throwable*, ma anche sorella della classe *Error*, che a sua volta è madre di un'intera gerarchia di oggetti "lanciabili". Che differenza c'è tra un'eccezione e un errore? In generale, le eccezioni sono problemi per i quali è concettualmente possibile trovare una soluzione a runtime; gli errori sono di solito problemi che che non ha senso cercare di "catchare", di solito perché dipendono dal sistema o dalla struttura del programma. Un errore è qualcosa che non si può correggere senza fermare il programma. Ad esempio, la Java Virtual Machine lancia un Error se incontra una classe compilata con una versione più recente e incompatibile del compilatore Java. È roba che riguarda la Virtual Machine, non il programmatore. Non dovresti mai definire né lanciare un *Error*. Un esempio di *Error* è illustrato nell'*Esercizio 1*. Qui termina la nostra miniserie di due articoli sulle eccezioni. Il mese prossimo ci avvicineremo alla conclusione del nostro corso con un argomento importantissimo: le collezioni. Non mancare.

Paolo Perrotta

Scrivi una classe A e una classe B. B contiene un main() che usa A:

```
public class B {
  public static void main(
              String[] args) {
    A a = new A(); }
}
```

Compila il tutto. Poi cancella il file A.class, e prova nuovamente ad eseguire B. Cosa ti aspetti che succeda?

NOTA

UNA BRUTTA FAMIGLIA

Se hai qualche esperienza di programmazione a oggetti, ti invito a fare un piccolo esercizio di design. Rileggi la struttura della gerarchia delle eccezioni (descritta nel primo paragrafo di questo articolo) e prova a giudicarne il design. Cosa ne pensi? Cosa cambieresti, se potessi tornare indietro nel tempo alla nascita del linguaggio?
Io ritengo che questa gerarchia sia stata concepita male. Non ha senso che le *RuntimeException* siano *"unchecked"* e tutte le altre eccezioni siano *"checked"*.
Le eccezioni *checked* hanno un comportamento più specifico di quelle *unckecked*, perché fanno tutto quello che fanno le eccezioni *checked* e in più ci costringono a dichiararle. Una sottoclasse non dovrebbe eliminare un comportamento specifico di una sua superclasse – al contrario, dovrebbe essere la sottoclasse a specializzare il comportamento della superclasse. Un indizio di questo problema sono frasi come:

"Se un'eccezione deriva da Exception ma non da RuntimeException, allora…".

Visto che ci sono osservo anche che il nome *RuntimeException* non ha molto senso, perché tutte le eccezioni (come qualsiasi altro oggetto) esistono solo a *runtime.* Sarebbe meglio se tutte le *Exception* fossero *unchecked* per default, ed esistesse una sottoclasse di *Exception,* chiamata forse *CheckedException,* che faccia da madre a tutte le eccezioni *checked.* Purtroppo non possiamo più cambiare questi aspetti fondamentali di Java, e ci tocca accontentarci di questa gerarchia bizzarra.

Import dei dati e generazioni di grafici statistici

Grafici in 3D

Descriveremo come realizzare dei grafici con il controllo MsChart, uno strumento flessibile che può ricevere dati da una matrice, una griglia o da una fonte di dati esterna

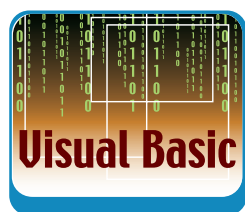
Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
Conoscenze di base del controllo DataGrid, degli Array

Software
Visual Basic 6 SP6

Impegno

Tempo di realizzazione

Il controllo *MSChart* permette di realizzare grafici di vario tipo (con istogrammi, linee, aree, ecc.) bi (2D) e tridimensionali (3D). I dati rappresentati con un *MSChart* possono essere recuperati da una fonte di dati esterna (database, foglio di calcolo, documento XML) o da un elemento strutturato (*Matrice, Griglia, RecordSet* ecc). L'obiettivo di questo articolo è descrivere come realizzare dei grafici e come modificarli in fase di esecuzione. Per questo, nel corso dell'articolo, sono presentate le principali caratteristiche del controllo *MSChart* ed alcune funzioni API che permettono di gestire i pulsanti della tastiera (in particolare il tasto *Ctrl* che è utilizzato per ruotare i grafici 3D).

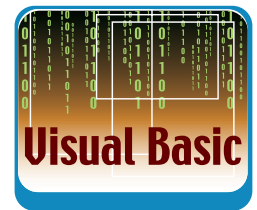
IL CONTROLLO MSCHART

Il controllo *MSChart* può essere usato per creare grafici statici o dinamici. Per realizzare le sue funzionalità poggia su diversi oggetti, tra i quali citiamo: *DataGrid,* utilizzato per immagazzinare e gestire le serie numeriche alla base dei grafici; *Axis* per impostare gli assi; *Backdrop* per impostare l'area dietro il grafico; *CategoryScale* per la scala degli assi; *View3D* per l'orientamento dei grafici tridimensionali; *Legend* per descrivere le serie numeriche; *DataPoints,* per descrivere gli attributi di un dato del grafico; *SeriesCollection* per collezionare degli oggetti *Series* (serie) cioè le serie dei dati; *Plot* per rappresentare l'aria in cui è visualizzato il grafico ecc. Le serie di dati del controllo *MSChart* possono essere recuperati e gestiti con diverse tecniche, per esempio con una matrice collegata al controllo tramite la proprietà *ChartData;* con l'oggetto *DataGrid* associato al controllo ed impostabile attraverso il metodo *SetData;* con una fonte dati esterna collegata al controllo attraverso la proprietà *DataSource.* Le caratteristiche dei grafici (intestazione, colore, etichette assi ecc.), possono essere impostate, attraverso la finestra delle proprietà, oppure utilizzando i metodi, gli eventi e le proprietà del controllo, collegati agli oggetti citati prima. Nel paragrafo successivo è presentato il primo esempio di grafico.

Fig. 1: ***Il grafico e la griglia dei dati***

IL PRIMO GRAFICO

Un modo semplice per creare dei grafici è quello di utilizzare delle serie di dati casuali. Creare un nuovo progetto, referenziare la *MSChart Control 6.0 (MSCHRT20.OCX)* e sul *form1* inserire un controllo *MSChart1* e un pulsante con il seguente codice:

```
Private Sub Command1_Click()
 Dim tipo As Integer
 Static numero As Integer
 tipo = Int(17 * Rnd)
 While tipo > 9 And tipo < 14 Or tipo = 15
  tipo = Int(17 * Rnd)
 Wend
 With MSChart1
  numero = numero + 1
  .RandomFill = True
  .ShowLegend = True
  .Legend.Location.LocationType = VtChLocationTypeTop
  .chartType = tipo
```

```
 .Title = "Random N° " + CStr(numero) + " - "
                                   + CStr(tipo)
 End With
End Sub
```

Con il codice precedente, ogni volta che si preme il pulsante *Command_1* è creato un grafico con il titolo *"Random - N° prova - tipo"* (impostato con proprietà *Title*), in uno degli stili disponibili (impostato con proprietà *chartType*). Nella procedura, oltre alla proprietà *RandomFill,* che carica la griglia con numeri casuali, sono utilizzate le proprietà che permettono di mostrare la legenda del grafico (nella parte superiore – *Top*) cioè *ShowLegend* e *Legend .Location.LocationType* (*Legend* è uno degli oggetti collegati alla *MSChart* citati in precedenza).

L'APPLICAZIONE D'ESEMPIO

La nostra applicazione consentirà di creare grafici con i dati contenuti in una griglia, in una matrice o in un foglio Excel (fonte dati esterna). I dati considerati sono le voci di un bilancio familiare. Questi dati, per semplificare, sono raggruppati per quadrimestre e secondo le seguenti categorie: *Affitto, Alimentari, Viaggi* e *spese Varie;* quindi, possono essere contenute in una griglia o in una matrice di 12 elementi (tre righe e quattro colonne). L'applicazione consente l'acquisizione dei dati, la manipolazione (creazione, modifica e ridimensionamento) dei grafici e l'interazione, con scambio di dati, tra *MSChart* e *MSHFlexGrid.* Nel progetto, dunque, oltre alla libreria *MSChart* bisogna referenziare la *MSHFlxGd.ocx* ed inserire due *Form,* uno per visualizzare il grafico *(Form1)* e l'altro per visualizzare i dati nella *MSHFlexGrid (Form2).* Sulla *Form1,* oltre al controllo *MSChart,* bisogna inserire un controllo *Menu* e un *PictureBox* con nell'interno un *Frame,* due *Label,* un *ComboBox* e un *CheckBox.* Il *ComboBox* (nominato *cmbTipoGraf)* consente di selezionare il tipo di grafico, il *CheckBox* (nominato *CheckCTRL*) consente d'impostare lo stato del tasto *Ctrl* (necessario per ruotare i grafici 3D), le label forniscono informazioni sui punti del grafico e sullo stato del tasto *Ctrl.* Il tasto *Ctrl* è controllato con due funzioni API: *Keybd_event* e *GetKeyState.* Il controllo *PictureBox,* posto sopra il controllo *MSChart1,* come si evince dalla **Figura 2**, è stato utilizzato per semplificare l'operazione di ridimensionamento della Form. Nel *Menu,* bisogna prevedere le voci per eseguire i seguenti comandi: disegna grafico utilizzando array, disegna grafico utilizzando griglia, mostra la griglia dei dati sul *Form2,* carica i dati del grafico da un file Excel. L'acquisizione dati da Excel e l'interazione con *MSHFlexGrid,* però, verrà descritta nel successivo appuntamento.

IMPOSTAZIONI E DICHIARAZIONI

Nella *Form1* (o in un modulo di servizio), innanzitutto, bisogna inserire la dichiarazione della *Matrice* utilizzata per passare i dati al controllo *MSChart.*

```
Const colonne = 4
Const righe = 3
Dim ArrDati(righe, colonne)
```

La costante *colonne* indica il numero delle serie di dati, mentre la costante *righe* indica il numero di valori di ogni serie. Da notare che quando i primi valori delle serie sono delle stringhe, dalla proprietà *ChartData,* vengono considerate come le etichette delle serie. La matrice è caricata attraverso la seguente procedura:

```
Sub CaricaDati()
 ArrDati(1, 1) = 495 'affitto
 ArrDati(2, 1) = 546
 ArrDati(3, 1) = 502
 ArrDati(1, 2) = 268  'alimentari
 ArrDati(2, 2) = 258
 ArrDati(3, 2) = 464
 ArrDati(1, 3) = 142  'viaggi
 ArrDati(2, 3) = 139
 ArrDati(3, 3) = 164
 ArrDati(1, 4) = 321 ' varie
 ArrDati(2, 4) = 156
 ArrDati(3, 4) = 168
End Sub
```

Fig. 2: Il menu dell'applicazione

Nella *Form_Load,* invece, s'impostano alcune caratteristiche del controllo *MSChart* e si avvia il caricamento del *cmbTipoGraf* e della matrice *ArrDati.*

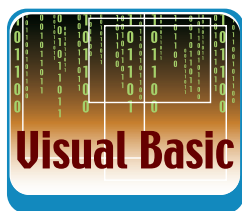
```
Sub Form_Load()
 With MSChart1
    .ShowLegend = True
    .AllowDynamicRotation = True
    .Legend.Location.LocationType =
                              VtChLocationTypeTop
    .Refresh
 End With
   CaricaDati
   CaricaCombo
End Sub
```

La procedura *CaricaCombo* carica il *cmbTipoGraf* con la descrizione dei tipi di grafici supportati dal controllo (controllare il box laterale).

```
Sub CaricaCombo()
 With cmbTipoGraf
  .AddItem "3d Barre"
  .AddItem "2d Barre"
  .AddItem "3d Linee"
```

ALCUNE PROPRIETÀ MSCHART

- **ShowLegend stabilisce se sarà visualizzata la legenda del grafico;**
- **AllowDynamicRotation imposta la rotazione dei grafici 3-D;**
- **ColumnLabelCount, RowLabelCount impostano il numero di livelli delle etichette di colonne e righe;**
- **ColumnLabel, RowLabel restituiscono o impostano le etichette di colonna e di riga;**
- **Column, Row impostano o restituiscono la colonna e la riga corrente della griglia;**
- **ColumnCount, RowCount impostano o restituiscono il numero di colonne e righe della griglia.**

```
    .AddItem "2d Linee"
    .AddItem "3d Area"
    .AddItem "2d Area"
    .AddItem "3d Istogrammi"
    .AddItem "2d Istogrammi"
    .AddItem "3d Combinazione"
    .AddItem "2d Combinazione"
    .AddItem "2d Torta"
    .AddItem "2d XY"
    .ListIndex = 1
  End With
End Sub
```

API

Per simulare la pressione di un tasto si può usare la funzione *Keybd_event*. Questa richiede i seguenti parametri: il virtual-key del tasto; un codice hardware; un flag che specifica vari aspetti del funzionamento operativo del tasto (pressione, rilascio del tasto ecc.) e un altro parametro che fornisce informazioni supplementari (negli esempi è stato posto a zero). Per stabilire lo stato di un tasto si può utilizzare la funzione *GetKeyState*. Essa come parametro ha un virtual-key (del tasto da controllare) e come risultato restituisce un intero.

Oltre alla *CaricaCombo* bisogna usare la *cmbTipoGraf_Click*, descritta sotto, per impostare un valore numerico nella proprietà *ChartType*.

```
Private Sub cmbTipoGraf_Click()
  Select Case cmbTipoGraf.ListIndex
    Case 0 To 9
      MSChart1.chartType = cmbTipoGraf.ListIndex
    Case 10
      MSChart1.chartType = VtChChartType2dPie
    Case 11
      MSChart1.chartType = VtChChartType2dXY
  End Select
  If Mid(cmbTipoGraf, 1, 2) = "3d" Then
    CheckCTRL.Visible = True
    Labelctrl = "Per ruotare il grafico " _
    & "tieni premuto Ctrl o clicca il CheckBox attiva Ctrl"
    Frametipo.Width = 4935
  Else
    CheckCTRL.Visible = False
    Labelctrl = ""
    Frametipo.Width = 3615
  End If
End Sub
```

Fig. 3: **Un grafico 3-d ruotato con il Mouse**

Nella procedura precedente, nel caso di grafico tridimensionale (3D), oltre ad impostare la proprietà *chartType*, vengono fatte le seguenti azioni: impostare la *labelctrl* (che fornisce informazioni sullo stato del tasto *Ctrl*), mostrare il *CheckCTRL* e ridimensionare il frame (il tutto in base agli elementi da mostrare).

La *cmbTipoGraf_Click* è eseguita anche al primo caricamento del Form grazie alla *ListIndex = 1* (presente nella *CaricaCombo*). Si fa notare che i valori numerici (3615 e 4935) impostati nella proprietà *Frametipo.Width* dipendono dalle dimensioni del form.

PASSAGGIO DATI CON MATRICE

Come accennato la prima voce del menu dell'applicazione è *Quadrimestre Array* che abilita la seguente procedura.

```
Private Sub MnuQua_Click()
    With MSChart1
     .ChartData = ArrDati
     .Title = "Spese per quadrimestre Array"
     .ColumnLabelCount = 1
     .Column = 1
     .ColumnLabel = "Affitto"
     .Column = 2
     .ColumnLabel = "Alimentari"
     .Column = 3
     .ColumnLabel = "Viaggi"
     .Column = 4
     .ColumnLabel = "Varie"
     .RowLabelCount = 1
     .Row = 1
     .RowLabel = "Primo"
     .Row = 2
     .RowLabel = "Secondo"
     .Row = 3
     .RowLabel = "Terzo"
     .Refresh
   End With
End Sub
```

Nella *MnuQua_Click* dopo aver caricato i dati (con *ChartData*) sono impostati il titolo del grafico e le label per le colonne *(ColumnLabel)* e per le righe *(RowLabel)*.

PASSAGGIO DATI CON GRIGLIA

Al controllo *MSChart* è associato un oggetto *DataGrid*, cioè una specie di matrice virtuale che contiene i dati del grafico e le etichette che descrivono le serie. Per gestire il *DataGrid* sono disponibili diversi metodi e proprietà. Di seguito presentiamo la procedura *MnuQuaGriglia_Click*, relativa alla voce di menu *Quadrimestre Griglia*.

Si fa notare che, per semplificare, i dati inseriti nella *Griglia* non sono letti da una fonte dati esterna, ma sono quelli di *ArrDati*.

```
Private Sub MnuQuaGriglia_Click()
 Dim Nullflag As Boolean
```

```
Dim colonna As Integer, riga As Integer
With MSChart1.DataGrid
 .columnCount = colonne
 .rowCount = righe
 For colonna = 1 To .columnCount
  For riga = 1 To .rowCount
   If (ArrDati(riga, colonna)) <> vbNull Then
    Nullflag = False
   Else
    Nullflag = True
   End If
   .SetData riga, colonna, _
    ArrDati(riga, colonna), _
    Nullflag
  Next riga
 Next colonna
 .ColumnLabel(1, 1) = "Affitto"
 .ColumnLabel(2, 1) = "Alimentari"
 .ColumnLabel(3, 1) = "Viaggi"
 .ColumnLabel(4, 1) = "Varie"
 .RowLabel(1, 1) = "Primo"
 .RowLabel(2, 1) = "Secondo"
 .RowLabel(3, 1) = "Terzo"
 MSChart1.Title = "Spese per quadrimestre Griglia"
End Sub
```

Nella *MnuQuaGriglia,* dopo aver impostato le dimensioni della griglia, carichiamo i dati, utilizzando il metodo *SetData* (*MSChart1.DataGrid.SetDat*), quest'ultimo ha la seguente sintassi:

```
SetData (riga, colonna, puntodati, flagnullo)
```

Riga e *colonna* identificano la posizione del "punto dati" sulla griglia, *puntodati* è il valore del punto del grafico; *flagnullo* è un Boolean che indica se il valore passato è nullo. Per leggere un valore dalla griglia, bisogna utilizzare il metodo *GetData,* che ha la seguente sintassi:

```
GetData (riga, colonna, puntodati, flagnullo).
```

Nella seconda parte della *MnuQuaGriglia* s'impostano le label delle colonne, con la proprietà *ColumnLabel(colonna, 1),* e quelle delle righe con *RowLabel(riga,1).* La sintassi di queste proprietà è del tipo:

```
ColumnLabel(colonna, indiceetichetta) [= testoetichetta]
```

Dove: *colonna* identifica una colonna di dati e *indiceetichetta* è il livello di etichetta considerato. Grazie ad *Indiceetichetta* ad una colonna (o ad una riga) è possibile associare più livelli di etichette, per esempio possiamo avere l'etichetta prodotti alimentari che raggruppa l'etichette *Pane, Pasta,* ecc. Il parametro testo etichetta è facoltativo dato che *ColumnLabel* può essere utilizzata anche per recuperare il valore di una etichetta.

RUOTARE I GRAFICI 3D

Come accennato un grafico 3D può essere ruotato, con il mouse, tenendo premuto, contemporaneamente, il tasto *Ctrl (Control).* A tal fine, sulla *Form1* è stato previsto il *CheckCTRL* per attivare il codice che permette di simulare la pressione del tasto *Ctrl.* Questo codice usa le seguenti routine API:

```
Private Declare Sub keybd_event Lib "user32" _
        (ByVal bVk As Byte, _ByVal bScan As Byte, ByVal
            dwFlags As Long, ByVal dwExtraInfo As Long)
Private Declare Function GetKeyState Lib "user32" _
                    (ByVal nVirtKey As Long) As Integer
```

Queste le costanti utilizzate:

```
Const KEYEVENTF_KEYUP = &H2
Const KEYEVENTF_EXTENDEDKEY = &H1
Const HK_CONTROL = &H11
```

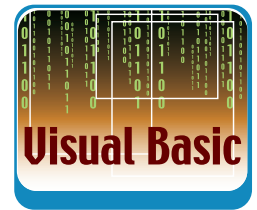
il codice è inserito nella *MSChart1_MouseMove*

```
Private Sub MSChart1_MouseMove(Button As Integer,
                 Shift As Integer, X As Single, Y As Single)
 If CheckCTRL.Value = 1 And Mid
                    (Me.cmbTipoGraf, 1, 2) = "3d" Then
  keybd_event HK_CONTROL, &H45,
                          KEYEVENTF_EXTENDEDKEY, 0
  `clicca tasto
 Else
  keybd_event HK_CONTROL, &H45, KEYEVENTF_
               EXTENDEDKEY _ Or KEYEVENTF_KEYUP, 0
  `disattiva tasto
 End If
End Sub
```

Per evitare che, dopo la chiusura della *Form1,* il *Ctrl* resti "premuto", nella *Form_Unload* bisogna inserire il seguente codice.

```
Private Sub Form_Unload(Cancel As Integer)
 If GetKeyState(HK_CONTROL) <> 0 Then
  keybd_event HK_CONTROL, &H45, KEYEVENTF_
                                    EXTENDEDKEY _
  Or KEYEVENTF_KEYUP, 0
    `disattiva tasto
 End If
End Sub
```

Infine, facciamo notare che, nella *Form_Unload,* si usa la *GetKeyState* per stabilire se il tasto *Ctrl* è "premuto". Il mese prossimo estenderemo l'applicazione importando i dati da un foglio Excel.

Massimo Autiero

VIRTUAL – KEY

Tra i parametri delle funzioni *GetKeyState* e *Keybd_Event* c'è un virtual-key, in particolare il virtual-key del tasto *Ctrl*. Ricordiamo che i virtual-key servono ad identificare, in modo univoco, i tasti indipendentemente dall'Hardware e dal Software. Il set di virtual-key è composto da 256 codici di un byte. Come per il codice ANSI e ASCII anche per i virtual-key è possibile recuperare su Internet o nella guida in linea di Visual Basic la tabella con tutti i codici.

Costruiamo un catalogo musicale di CD rapidamente

Musica on-line in nove tempi

Utilizziamo DaDaBIK, un software scritto in PHP, per creare una comoda interfaccia Web verso un database MySQL che conterrà un archivio musicale

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

In questo numero creeremo un'applicazione per gestire un catalogo musicale on-line. Il database sarà MySQL, il linguaggio: PHP. La novità sarà che non scriveremo una sola riga di codice, a fare tutto per noi sarà DaDaBIK.

DI COSA ABBIAMO BISOGNO

Prima di tutto un database denomianto *music_online_db*. Lo schema del database lo trovate in **Figura 1**, contiene quattro tabelle: *albums_tab*, *genres-_tab*, *customers_ tab*, *cities_tab*.
Successivamente ci serve DaDaBIK. Lo utilizzeremo per realizzare l'applicazione.
Infine PHP, MySQL e un Web Server che supporti il tutto. Se non avete un vostro sito on-line potete provare in locale utilizzando un sistema preconfigurato come WAMP. Lo trovate nel nostro CD-Rom.

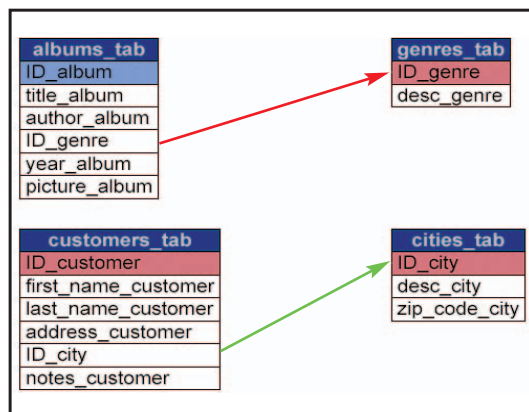



Fig. 1: **Lo schema delle relazioni e delle tabelle del Database music_online_db**

Nel CD allegato alla rivista e/o sul sito web di ioProgrammo *www.ioprogrammo.it* viene proposta la query SQL necessaria alla creazione delle quattro tabelle

IL SOFTWARE GRATUITO DADABIK

È possibile scaricare DaDaBIK dal sito www.dadabik.org, il software è rilasciato sotto la licenza GNU GPL e, al momento in cui l'articolo andrà in stampa, dovrebbe essere disponibile la versione 3.1. È possibile utilizzare DaDaBIK con qualsiasi Web server che supporti PHP, è inoltre indispensabile avere installato MySQL. DaDaBIK non è un tool di amministrazione, non è quindi possibile creare nuovi database, nuove tabelle o modificare i nomi dei campi, viene invece fornita un'interfaccia per agire sui contenuti delle tabelle.

INSTALLAZIONE DI DADABIK

Ponendo che MySQL e il Web server girino in locale e che la password di root per MySQL sia *rtpwd*, i passi da seguire sono i seguenti:

1) Creare una directory sotto la root del Web server, chiamata ad es. *dadabik*

2) Scompattare il file .zip di DaDaBIK e copiare il contenuto della cartella *program_files* nella cartella poc'anzi creata.

CONNETTIAMOCI AL DATABASE

Editiamo il file *config.php* (situato nella cartella *include*) e modifichiamo i parametri *$host, $user, $pass, $db_name* e *$site_url;* nel nostro caso, ecco come si presenterà il file *config.php*

```
$host = 'localhost';
$db_name = 'music_online_db';
$user = 'root';
$pass = 'rtpwd';
$site_url = 'http://localhost/dadabik/';
```

Avviare la pagina *http://localhost/dadabik/install .php* per procedere all'installazione e la pagina *http://localhost/dadabik/index.php* per iniziare ad utilizzare il software.

SCEGLIAMO I CAMPI

A questo punto dovremo scegliere quali sono i campi che serviranno per far funzionare la nostra applicazione.

Dall'interfaccia di amministrazione (*http://localhost/dadabik/admin.php*), è possibile iniziare a customizzare DaDaBIK, in particolare, cliccando su *"Interface configurator"* è possibile accedere a una sottosezione che permette di settare:

- l'etichetta dei campi
- in quali form ciascun campo deve comparire
- quali campi sono *required*
- il tipo di input control da utilizzare (una textbox, un drop-down menu, un date control….)
- il contenuto del campo (*numeric, alphanumeric, email, url, generic_file, image_file…*)
- le relazioni tra i campi delle diverse tabelle.

COLLEGHIAMO I CAMPI

Il nostro Database è strutturato in modo che l'elenco dei generi musicali sia contenuto nella tabella *genres_tab*. I record della tabella *albums_tab* che contiene le informazioni sugli album musicali, non conterranno una stringa per descrivere il genere ma semplicemente l'ID del genere musicale descritto nella tabella *genres_tab*. Dobbiamo istruire DaDaBIK a gestire questa situazione. Il nostro scopo è quello di dare all'utente la possibiltà, in fase di inserimento e modifica di un album, di avere un menù che elenca la descrizione di tutti i generi disponibili (prelevati da *genres_tab*). Allo scopo utilizziamo l'*Interface Configurator:* posizioniamoci sul campo *ID_ genre* (tabella *albums_ tab*) e scegliamo:

select_single	come *Field type* (questo visualizzerà il menu a tendina)
ID_genre	come *Primary key field*
genres_tab	come *Primary key table*
desc_genre	come *Linked fields*

In questo modo è come se impartissimo dei comandi del tipo: "il campo *ID_genre* (tabella *genres_tab*) va visualizzato in fase di inserimento/modifica/ricerca tramite un menu a tendina (*select_ single*), il menu a tendina va riempito con tutti i record di genres_tab (*primary key table*), utilizzando *desc_genre* (*linked field*) come descrizione e *ID_genre* (primary *key field*) come *value* (ciò che verrà di fatto inserito/ricercato nei record)."

LA GESTIONE DELLE COPERTINE

Immaginiamo di voler associare, e successivamente visualizzare, per ogni album la copertina. Vogliamo che in fase di inserimento/modifica l'utente possa scegliere dal proprio PC locale il file (gif, png o jpeg) da uploadare. L'immagine deve essere salvata in una directory prescelta, mentre nel DB viene registrato il solo nome del file. Procediamo scegliendo:

image_file	come *Field type* (identifica il campo come file, e in particolare come immagine)

Successivamente sarà necessario modificare il file *config.php* in modo da specificare i parametri *$upload_relative_url* e *$upload_directory*, che indicano dove vogliamo salvare i file uploadati. Per esempio:

```
$upload_relative_url = 'uploads/';
$upload_directory = 'e:\\web\\dadabik\\uploads\\';
```

Da notare i doppi back slash (il primo è un carattere di escape per il secondo); se installassimo DaDaBIK su un server unix-like il path sarebbe stato qualcosa del tipo:

```
$upload_directory = /home/web/dadabik/uploads/';
```

È necessario che l'utente loggato al Web server (nel caso di accesso anonimo ad es. *Internet guest account* per MS IIS e *nobody* per Apache) abbia i permessi di scrittura sulla cartella *uploads*.
È inoltre necessario abilitare da *config.php* le estensioni di file che vogliamo accettare in upload,

NOTA

CHIAVI PRIMARIE

È bene ricordare che, per sfruttare tutte le features di DaDaBIK è necessario che ogni tabella da gestire abbia una chiave primaria associata.

PHP

È un linguaggio di programmazione utilizzato soprattutto per il Web developement. I suoi punti di forza, oltre alla licenza open source con la quale è rilasciato, sono la velocità di esecuzione, la semplicità di apprendimento, la disponibilità di una quantità enorme di funzioni per svolgere i compiti più vari e il fatto che sia multipiattaforma. La sua popolarità è cresciuta moltissimo negli ultimi anni.

SUL WEB

DaDaBIK:
www.dadabik.org
PHP:
/www.php.net
MySQL:
/www.mysql.com
htmlArea:
www.interactivetools.com/products/htmlarea
PHPMyAdmin:
www.phpmyadmin.net
PHPNuke:
www.phpnuke.org

nel nostro caso:

```
$allowed_file_exts_ar[0] = 'jpg';
$allowed_file_exts_ar[1] = 'gif';
$allowed_file_exts_ar[2] = 'png';
```

ELIMINIAMO I DOPPIONI

Possiamo attivare un controllo sui record duplicati per singola tabella. Ad esempio sulle tabelle *customers tab*, campo *last_name_customer*, assicuriamoci che sia settato:

Check for duplicated entries during insert? Yes

In aggiunta, DaDaBIK valuta la "somiglianza" tra le parole e segnala, quindi, un "sospetto" record già esistente; è tuttavia possibile "scegliere" quanto devono essere simili i contenuti dei campi per essere valutati come possibili duplicati; si modificando *$percentage_ similarity*, settato nel file *config.php* indica un valore percentuale (se impostato a 100% le stringhe devono essere uguali).

CONFIGURIAMO LA RICERCA

Per ogni campo che vogliamo includere nel form di ricerca è necessario settare: *Field present in the search form? Yes.* Inoltre, per ogni campo incluso, possiamo specificare quali operatori di ricerca l'utente potrà scegliere tra: *is_equal*, *contains*, *starts_with*, *ends_with*, *greater_ than* e *less_than*, corrispondenti ai classici operatori *SQL: LIKE '%….%', LIKE '….%', LIKE '%….';*> e <. I risultati della ricerca possono essere esportati in un file CSV cliccando l'apposito link presente nella pagina dei risultati.

PERSONALIZZARE LA GRAFICA

Vediamo, infine, come includere il progetto DaDaBIK in un sito o in una Web application adeguandosi al layout preesistente. Il primo passo necessario è quello di modificare i file *header.php* e *footer.php* (directory *include*), che contengono il codice HTML che "apre" e "chiude" tutte le pagine visualizzate; nei due file dovremo inserire il codice HTML necessario per rendere il layout di DaDaBIK compatibile con quello preesistente, operazione che risulta particolarmente semplice specialmente se il sito "contenitore" prevede anch'esso una struttura a *header* e *footer*, come spesso accade. Ad esempio, per inserire il menu generale del sito "contenitore" nella parte sinistra, sarà necessario "aprire" una cella HTML supplementare contenente i vari link alle pagine del sito. Il secondo passo è la modifica dei fogli di stile, in questo modo possiamo modificare il tipo di font utilizzato, la dimensione, i colori e molti altri dettagli. DaDaBIK dispone di due file: *styles_screen.css* e *style_print.css* (directory *css*), il primo destinato a controllare gli stili a video; il secondo dedicato alla stampa. Infine, modificando opportunamente il file *config .php* potremmo selezionare una delle sei lingue (italiano, inglese, tedesco, olandese, spagnolo e francese) con cui vorremmo che DaDaBIK "parlasse": *$language = 'italian';*

Eugenio Tacchini

LE OPERAZIONI CHE SI POSSONO COMPIERE SUI RECORD

Dopo la fase di installazione, per ogni singola tabella sarà possibile:

- **Visualizzare tutti i record, con possibilità di ordinarli cliccando sul titolo della colonna corrispondente; i record vengono suddivisi automaticamente su più pagine (paginazione)**
- **Effettuare una ricerca, sulla base di uno o più campi della tabella, impostando la condizione di ricerca ("inizia con", "contiene", "maggiore", "minore"…) e l'operatore booleano che lega le condizioni (and,or)**
- **Visualizzare il dettaglio di un record**
- **Inserire, modificare o esportare (in formato CSV) i record**

I TIPI DI DATO GESTITI DA DADABIK

Negli esempi finora esaminati abbiamo utilizzato i tipi di campi *text*, *select_single* e *image_file*; tuttavia DaDaBIK gestisce diversi altri tipi: *textarea*, *password*, *rich_editor*, *insert_date*, *update_date*, *date*, *generic_file* (si comporta come *image_file* ma in fase di visualizzazione mostra semplicemente un link al file uploadato), *ID_user* (un campo in cui DaDaBIK inserisce automaticamente lo username dell'utente corrente al momento in cui il record viene inserito).

IL CONTENUTO DEI CAMPI

Il tipo di campo influenza il modo in cui i contenuti stessi vengono visualizzati e i controlli effettuati in fase di inserimento/ modifica. Abbiamo a disposizione:

- **alphabetic:** *visualizzazione normale, solo caratteri alfabetici ammessi*
- **alphanumeric:** *visualizzazione normale, ammessi tutti i caratteri*
- **numeric:** *visualizzazione normale, ammessi solo caratteri numerici*
- **url:** *il contenuto viene visualizzato come link, ammesse solo URL sintatticamente valide*
- **email:** *il contenuto viene visualizzato come link mailto, ammessi solo indirizzi e-mail validi*
- **phone:** *visualizzazione normale, ammesse stringhe composte dal carattere + seguito da caratteri numerici*
- **html:** *visualizzazione del risultato HTML (negli altri casi i tag HTML vengono trasformati in apposite HTML entities), ammessi tutti i caratteri*

BluePrints: i pattern "naturali" di J2EE

Un motore di ricerca in Java parte prima

Iniziamo la progettazione di un motore di ricerca, sfruttando i design pattern che vengono descritti nei Java BluePrints per J2EE In questa prima parte parleremo di MVC, View Helper e altro

Lo scopo di questa serie di due articoli è quello di fornire informazioni utili per il corretto utilizzo degli strumenti offertici da J2EE. Tali informazioni provengono direttamente dai *BluePrints* della Sun, i documenti elettronici liberamente scaricabili che definiscono tra l'altro le *Best Practices* ed esempi di codice per J2EE. Con questo primo articolo, saremo in grado di implementare soluzioni basate su alcuni dei più importanti design pattern per J2EE. Dopo aver introdotto il pattern architetturale MVC, descriveremo i pattern *View Helper, Service Locator* e *Business Delegate*. Ci limiteremo così al "lato WEB" di J2EE (relativo alle tecnologie Servlet e JSP).
Nel prossimo articolo ci dedicheremo alla descrizione di pattern "lato Business" (relativo alla tecnologia EJB). Per apprezzare in pratica l'utilità di una progettazione basata sui pattern di J2EE, realizzeremo un semplice motore di ricerca.

Fig. 1: Uno screenshot del motore di ricerca che implementeremo

ARCHITETTURA MODEL 1

Per valutare i vantaggi di J2EE è dunque necessario non limitarsi ad utilizzare solo la tecnologia più semplice. Per esempio, per costruire il nostro motore di ricerca, potremmo limitarci al solo utilizzo di JSP, appunto la tecnologia apparentemente più semplice. Infondo. Una pagina JSP, può fare tutto quello che può fare una pagina ASP, con performance superiori e con a disposizione strumenti più potenti (forniti dalle classi Java). Ma ci sono vari limiti ad un'architettura basata solo su JSP (che viene detta architettura *Model 1*). Con il *Model 1*, infatti, le varie pagine JSP, oltre a gestire la logica di presentazione tramite tag generati dinamicamente, devono anche gestire la logica di business, per esempio creando algoritmi ed interrogando database. La natura di una JSP (che è costituita da tag), è quella di gestire la logica di presentazione delle pagine. L'utilizzo di classi Java a cui delegare la logica di business migliorerebbe la situazione, ma esiste un'alternativa più affidabile: l'architettura *Model - View - Controller* (MVC).

ARCHITETTURA MVC

L'MVC, definisce la separazione tra i componenti software che implementano il modello delle funzionalità di business, dai componenti che implemen-

Utilizza questo spazio per le tue annotazioni

tano la logica di presentazione e di controllo di tali funzionalità. Vengono quindi definiti tre tipologie di componenti che soddisfano tali requisiti:

- il **Model**, che implementa le funzionalità di business
- la **View**: che implementa la logica di presentazione
- il **Controller**: che implementa la logica di controllo.

Per poter applicare correttamente l'MVC, è necessario definire le responsabilità dei tre componenti che vengono evidenziate in **Figura 2**.

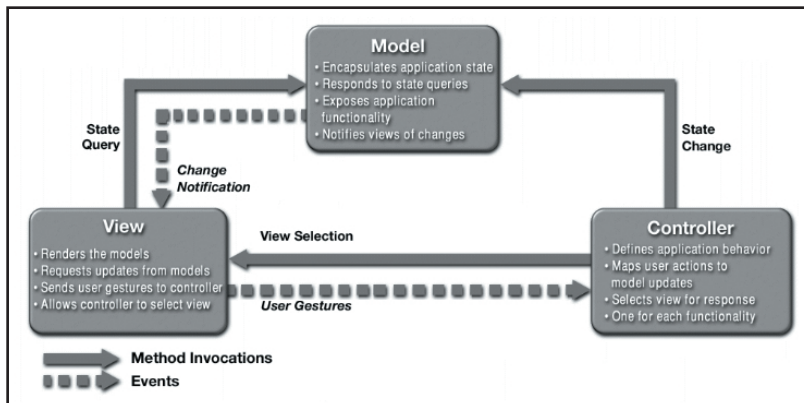



Fig. 2: Diagramma di interazione, che evidenzia le responsabilità dei tre componenti

- **Model**: analizzando la **Figura 2**, si evince che il core del software viene implementato dal *Model*. Questo, incapsulando lo stato dell'applicazione, definisce i dati e le operazioni che possono essere eseguite su di essi. Quindi, definisce le regole di business per l'interazione con i dati ed espone alla View ed al Controller rispettivamente le funzionalità per l'accesso e l'aggiornamento. In pratica, il *Model* "è" l'applicazione, ed espone alla *View* ed al *Controller* i metodi da chiamare, senza mostrare l'implementazione interna.
- **View:** la logica di presentazione dei dati viene gestita unicamente dalla *View*. Ciò implica che questa deve fondamentalmente gestire la costruzione dell' interfaccia grafica (GUI) che rappresenta il mezzo mediante il quale gli utenti utilizzeranno il sistema. Ogni GUI può essere costituita da schermate diverse che presentano più modi di interagire con i dati dell'applicazione. Infine, la *View* delega al *Controller* sia l'esecuzione dei processi richiesti dall'utente (dopo averne catturato gli input), sia la scelta delle eventuali schermate da presentare.
- **Controller:** questo componente ha la "responsabilità" di trasformare le interazioni dell'utente con la *View*, in azioni eseguite dal *Model*. Ma il *Controller* non rappresenta un semplice "ponte" tra *View* e *Model* come pensano in molti.

NOTA

I Java BluePrints definiscono un modello di programmazione applicativo per soluzioni end-to-end tramite l'utilizzo della piattaforma Java 2 Platform, Enterprise Edition (J2EE). Essi contengono linee guida, patterns, e codice per scenari reali della programmazione e della progettazione, che permettono la realizzazione di soluzioni robuste, scalabili, e portabili.

Il *Controller* implementa la "logica di controllo" dell'applicazione in quanto:

- realizza la mappatura tra input dell'utente e processi eseguiti dal *Model*
- seleziona le schermate della *View* richiesta.

Ciò significa che si può cambiare il modo in cui si interagisce con l'applicazione *(Model)*, semplicemente sfruttando *Controller* diversi.

ARCHITETTURE MODEL 2 E J2EE

Ma cosa sono il *Model*, la *View* e il *Controller?* Sono classi, metodi, librerie, applicazioni indipendenti, o cos'altro? La risposta è "dipende". Il pattern MVC governa le relazioni che intercorrono tra questi componenti, ma queste relazioni non dipendono dalla implementazione dei componenti stessi. Nel caso della architettura J2EE, si può dire che il *Model*, la *View* ed il *Controller* siano implementate non da classi, package o librerie, ma da tre tecnologie: rispettivamente *EJB*, *JSP* e *Servlet*. Le *Servlet* possono essere utilizzate per implementare la logica di controllo definendo l'esecuzione di un metodo. Infine, la tecnologia *EJB*, mettendo a disposizione tre tipologie di bean, permette di implementare la logica di business.

IN PRATICA: IL MOTORE

Scriveremo una semplice applicazione che funge da motore di ricerca utilizzando J2EE. Il codice dell'applicazione completa si trova nel CD allegato a questo numero di ioProgrammo. In queste pagine troverete solo i frammenti di codice più interessanti, ai fini delle nostre osservazioni. In particolare, saranno mostrati i frammenti di codice fondamentali che seguono l'esecuzione di una richiesta di ricerca da parte di un eventuale client. Per avere un'idea del flusso che dobbiamo seguire viene fornito un sequence diagram semplificato, di alto livello. Analizziamo il diagramma: l'utente fa una richiesta tramite un form messo a disposizione da una JSP. La richiesta viene inoltrata ad una servlet che interpretandola sceglie ed invoca un metodo di un EJB (*SearchEngineEJB*). Il *model*, quindi, applica la logica di business, interroga un DB e restituisce un risultato alla servlet. Quest'ultima in qualità di controller, controlla il risultato, e delega la sua presentazione alla giusta *view*. La pagina JSP *(SearchEngineFormView)*, contiene un form che rimanda alla servlet di controllo *(SearchEngineContoller)*, un parametro con la stringa da utilizzare per la ricerca, ed un parametro per sapere dove effettuare la ricerca

(se nel web o nei siti italiani). La servlet, fungendo da controller, ha la responsabilità di:

1. controllare la correttezza dei dati
2. indirizzare alla giusta funzionalità del model la richiesta dell'utente
3. scegliere la giusta *View* da presentare, fornendo ad essa i risultati restituiti dal *Model*

Il seguente metodo *doPost()* della servlet adempie facilmente alla prima responsabilità:

```
String search_text = request.getParameter(
                                  "search_text");
try {
  if (search_text == null || search_text.equals("")) {
    response.sendRedirect(instructionsViewAddress);
    return; }
...
```

Per quanto riguarda la seconda responsabilità si realizza mediante codice un po' più complesso:

```
  //Ottenimento dell'EJB object
  InitialContext ic = new InitialContext();
  String jndiHomeName = context.getInitParameter(
                              "jndi_ejb_home_name");
  Object objref = ic.lookup(jndiHomeName);
  SearchEngineEJBHome home =
      (SearchEngineEJBHome) PortableRemoteObject.
          narrow(objref, SearchEngineEJBHome.class);
  SearchEngineEJB searchEngineEJB = home.create();
  // EJB object ottenuto
  // Scelta della pagina dove verrà reindirizzato il
                              risultato della ricerca
  String search_type = request.getParameter(
                                  "search_type");
  Collection result = null;
  if (search_type != null &&
              search_type.equals("standard")) {
    result = searchEngineEJB.standardSearch(
                                  search_text);}
  else if (search_type != null &&
              search_type.equals("country")) {
    result = searchEngineEJB.countrySearch(
                                  search_text);}
  else { throw new SearchEngineException("Tipo di
          ricerca non consentito!" + search_type);}
  // gestione delle eccezioni
...
```

La terza responsabilità, si può codificare inserendo come attributo nello scope request il risultato ottenuto dall'invocazione del metodo remoto, e reindirizzando il flusso alla JSP, che si deve preoccupare della presentazione dei risultati. Quindi, la terza responsabilità viene implementata dalla servlet tramite il seguente codice:

```
  request.setAttribute("result",result);
  RequestDispatcher dispatcher = getServletContext(
              ).getRequestDispatcher(resultViewAddress);
    dispatcher.forward(request, response);
...
```

Tenendo conto che *result* è una *Collection* di oggetti di tipo *Site* (classe opportunamente creata per astrarre il concetto di sito web), la JSP può presentare i risultati in questo modo:

```
        <table>
        ...
          <td align="left">
            <% Collection result =
                        request.getAttribute("result");
                for (Iterator iter = result.iterator();
                                  iter.hasNext(); ) {
                Site site = (Site)iter.next(); %>
              <br>
              <% out.println("<a href="" + site.getAddress()
                              + "'>" + site.getTitle() +
            "</A><br>" + site.getAddress() +
              " <IMG SRC='img/"+ site.getLanguage()
                      +".gif' align='absmiddle'/>";
              <br> <% } %> </td>
        ...
```

Fig. 3: ***sequence diagram, interazioni tra componenti semplificata***

L'architettura MVC, non è complicata a livello logico, ma il codice che ci ha portato a generare non è proprio banale. In queste poche righe, infatti, si possono notare un paio di passaggi complicati e poco leggibili. Soprattutto, JSP alterna troppi *scriptlet* (codice Java) a semplici tag HTML.

IL PATTERN VIEW HELPER

L'applicazione del pattern *View Helper* può rendere più semplice e leggibile il lavoro di codifica della JSP per la presentazione dei dati. In pratica, un *View Helper* è una classe a cui la view delega la responsabilità di "adattare i dati", in modo tale che si abbia la possibilità di utilizzarli con semplicità. Ciò favorisce

J2EE BluePrints design patterns:
http://java.sun.com/blueprints/patterns/catalog.html

Il libro più venduto sui pattern J2EE:
http://java.sun.com/blueprints/corej2eepatterns/index.html

Il pattern MVC "vero":
www.claudiodesio.com/ooa&d/mvc.htm

facilità nella manutenzione, facilità nell'accomodare cambiamenti e leggibilità. Una buona classe candidata a diventare *View Helper* nella nostra applicazione, è la stessa classe *Site*. Si tratta solo di migliorare una classe che *View Helper* lo è già, fornendo sotto forma di stringa tutti i dati di cui ha bisogno la *View*. Ma aggiungendo nella classe *Site* i seguenti metodi:

```
public String getImageLanguageAddress() {
  return "img/"+getLanguage()+".gif";}
public String toString() {
  return "<A HREF='" + getAddress() + "'>" + getTitle()
        + "</A><br>" + getAddress() + " <IMG SRC='" +
  getImageLanguageAddress()+"' align='absmiddle'/>"; }
```

La JSP potrà presentare i dati con maggior semplicità, nel seguente modo:

```
<jsp:useBean id="result" scope="request"
                        class="java.util.Collection" />
<table>
...
  <td align="left">
    <% for (Iterator iter = result.iterator();
                              iter.hasNext(); ) {
    %>
    <br>
    <%= iter.next() %> <%-- l'invocazione a
                          toString() è automatica %>
    <br><% } %></td>
```

Se volessimo cambiare la presentazione di ogni sito, basterebbe cambiare il metodo *toString()*. Se volessimo invece modificare la struttura della tabella che contiene tutti i risultati della ricerca, basterebbero semplici modifiche alla JSP.

BUSINESS DELEGATE

Un altro frammento di codice che appare più complicato è quello che assolve alla seconda responsabilità della servlet controller. In pratica, si deve fare il *lookup()* ed il *narrow()* dell'home object dell'EJB remoto, si ottiene l'EJB object remoto mediante il metodo *create()*, si invoca il metodo di business appropriato, e si gestiscono le eccezioni. È ovviamente lecito codificare il tutto nella nostra servlet, ma non si può dire di aver fatto una buona scelta progettuale. La soluzione semplice, elegante e funzionale consiste nell'implementare il pattern noto come *"Business Delegate"*. Si costruisce a tale scopo una classe ad hoc intermedia (detta appunto *Business Delegate*), che fornisce un'interfaccia pubblica del tutto simile a quella offerta dagli EJB che si vogliono utilizzare. Questa classe incapsula eventuali parti di codice che possono essere estratti dalla nostra servlet, come l'accesso all'EJB remoto e la gestione delle eccezioni. Il nostro *Business Delegate* potrebbe essere così implementato:

```
...
public Collection standardSearch(String searchText)
                                                throws
    SearchEngineException, RemoteException {
  SearchEngineEJB searchEngineEJB =
                                    getSearchEngineEJB();
  try {
    return searchEngineEJB.standardSearch(searchText); }
  catch (RemoteException ex) {
    ex.printStackTrace();
    return new Vector();} }
public Collection countrySearch(String searchText) throws
    SearchEngineException, RemoteException {
  SearchEngineEJB searchEngineEJB =
                                    getSearchEngineEJB();
  try {
    return searchEngineEJB.countrySearch(searchText);}
  catch (RemoteException ex) {
        ex.printStackTrace();
    return new Vector(); } }
```

Di conseguenza la servlet potrebbe definire un codice più leggero come il seguente:

```
...
    String search_type = request.getParameter(
                                          "search_type");
    Collection result = null;
    if (search_type != null &&
                search_type.equals("standard")) {
      result = delegate.standardSearch(search_text);}
    else if (search_type != null &&
                search_type.equals("country")) {
      result = delegate.countrySearch(search_text);}
...
```

Le conseguenze sono tante:

1) riduzione dell'accoppiamento tra lo strato Web e lo strato EJB.
2) maggiore manutenibilità con una gestione per accomodare i cambiamenti centralizzata nel Business Delegate.
3) il Business Delegate può implementare la gestione delle eccezioni provando per esempio azioni di recovery, e nascondere eccezioni applicative provenienti dallo strato EJB, alla vista del client. Inoltre potrebbe sincronizzare i propri metodi, evitando problemi alla servlet che invoca i suoi metodi.
4) La prima conseguenza che può essere considerata negativa è che introduce uno strato in più, aggiungendo complessità alla nostra applicazione.
5) Bisogna stare attenti a non dimenticare che il

COESIONE E ACCOPPIAMENTO

L'accoppiamento (in inglese "coupling"), nella teoria OO è un metrica che misura la interdipendenza tra moduli (come classi, package etc...). Insieme ad un'altra metrica la coesione (in inglese "cohesion") permette di misurare la qualità del software: più è basso l'accoppiamento tra moduli e più è alta la coesione all'interno dei moduli, maggiore è la qualità del software. La coesione è la metrica con cui si misura quanto uno o più moduli software siano creati per assolvere ad un determinato compito.

Business Delegate è un proxy lato client, e che quindi tante invocazioni di metodi implicano traffico di rete. La semplicità dell'interfaccia del Business Delegate induce a considerare i servizi remoti come se fossero locali.

6) Può implementare meccanismi di *caching* al fine di migliorare le performance, e fornisce al client un'interfaccia semplice e funzionale ad hoc.

Per quanto riguarda quest'ultimo punto, introduciamo un altro pattern che può ulteriormente ottimizzare il nostro codice.

SERVICE LOCATOR

L'accesso all'home object mediante le chiamate a *lookup()* e *narrow()* è un'operazione che viene fatta più volte all'interno di un'applicazione. È facile centralizzare il tutto, quindi, con un metodo come il seguente:

```
public EJBHome getRemoteHome(String
                jndiHomeName, Class className) throws
    SearchEngineException {
  EJBHome home = null;
  if (serviceLocatorMap.containsKey(jndiHomeName)) {
    return (EJBHome) serviceLocatorMap.get(
                                jndiHomeName);}
  try { Object objref = ic.lookup(jndiHomeName);
        Object obj = PortableRemoteObject.narrow(
                                objref, className);
      home = (EJBHome) obj;
      if (home!=null) serviceLocatorMap.put(
                        jndiHomeName, home); }
      catch (NamingException ne) {
          throw new SearchEngineException(ne);}
      catch (Exception e) {
          throw new SearchEngineException(e); }
      return home; }
```

Una *hashmap (serviceLocatorMap)* viene interrogata ogni volta venga richiesto un home object. Se l'oggetto richiesto è presente nella hashmap, viene ritornato, altrimenti viene recuperato mediante le solite invocazioni a *lookup()* e *narrow()*, ed aggiunto alla *hashmap*. Questo meccanismo è un notevole vantaggio per la performance del software. Infatti, è inutile avere tanti home object per ottenere tanti EJB object dello stesso tipo, ne basta uno! Il *Service Locator* è una classe implementata come singleton (cioè istanziabile una sola volta) che può fare il lookup di tutte le risorse remote (come gli home object degli oggetti remoti) e mantenerle in cache . Questo discorso vale anche per altre risorse come *DataSource, Code, Topic, Connection, ConnectionFactory* etc… (ovvero tutto ciò che è mappabile con nomi JNDI), il che rende consigliabile l'implementazione di un Service Locator anche nello strato EJB. Se il Service Locator mette a disposizione un metodo come *getRemoteHome*, allora il codice del metodo del delegate si può ridurre a:

```
protected SearchEngineEJB getSearchEngineEJB()
                    throws SearchEngineException {
  String jndiName = "search_engine_ejb";
  try {
    SearchEngineEJBHome searchEngineEJBHome =
                          (SearchEngineEJBHome)
      ServiceLocator.
      getInstance().getRemoteHome(jndiName,
                    SearchEngineEJBHome.class);
    searchEngineEJB = searchEngineEJBHome.create();}
  catch (Exception e) {
    throw new SearchEngineException(
      "SearchEngineDelegate-->Unable to create
              remote object SearchEngineEJB", e);}
  return searchEngineEJB;}
public Collection standardSearch(String searchText) throws
    SearchEngineException, RemoteException {
  SearchEngineEJB searchEngineEJB =
                          getSearchEngineEJB();
  try {
    return searchEngineEJB.standardSearch(searchText);
…
```

E così tutti gli altri componenti dell'applicazione, che vogliono accedere agli oggetti distribuiti.

CONCLUSIONI

A questo punto abbiamo realizzato un'applicazione sfruttando alcuni pattern di J2EE. L'applicazione di ognuno di essi ha permesso di migliorare il nostro codice in modo considerevole. L'MVC, ha definito le regole di interazione dei componenti dell'applicazione. Il *Value Helper* ha permesso alla view che doveva presentare i risultati di una ricerca, di poter essere codificata in modo molto coinciso ed efficace. Il pattern *Business Delegate* invece, ha semplificato il codice della servlet controller. Ha infatti, fornito ad essa un'interfaccia semplificata, del tutto simile a quella remota che offre lo strato EJB. Inoltre ha incapsulato un meccanismo di delega proprio allo strato EJB, in maniera trasparente. Infine abbiamo, implementato un semplice *ServiceLocator*, che ci ha permesso di evitare eventuali duplicazioni di codice complesso, come quello che serve per ottenere un *home object*. Inoltre ha migliorato le prestazioni della nostra applicazione implementando un semplice ma efficace meccanismo di caching degli *home object*. Nel prossimo articolo esploreremo lo strato EJB ed altri interessanti pattern.

Claudio De Sio Cesari

Per compilare e deployare il progetto, si consiglia di utilizzare un IDE come JBuilder X o Eclipse (con plug-in Lomboz). Come application server, è possibile scegliere tra più alternative: Websphere Server, Weblogic Server o la coppia Tomcat – JBoss…

L'AUTORE

Claudio De Sio, è un consulente free-lance che si occupa di tecnologia Java, analisi, progettazione e architetture object oriented. Laureato in matematica, dal 1999 collabora con *Sun Educational Services* (come docente e mentore), ed altri importanti aziende dell'*IT*. Sul suo sito www.claudiodesio.com, ha pubblicato diverse risorse didattiche gratuite, relative agli ambiti dove è specializzato.

Tecniche algoritmiche per lo studio di fenomeni naturali

Sistemi caotici

Descriveremo dei semplici algoritmi per l'ottenimento di figure frattali capaci di imitare la forma di semplici forma di vita. Scopriremo le regole che si nascondono dietro l'apparente caos della vita

Utilizza questo spazio per le tue annotazioni

Fino a pochi anni fa, sarebbe stato impensabile creare un modello dei fenomeni naturali più complessi. La matematica tradizionale anche mediante sofisticate espressioni algebriche o sistemi di esse è riuscita ha descrivere molti dei fenomeni che si presentano in natura. Si è però sempre trattato di eventi che coinvolgevano punti o sistemi di essi, ossia oggetti. Così, con le conoscenze acquisite fino a qualche decennio fa (che da un lato attingevano alle nozioni geometriche euclidee e dall'altro si riferivano a strutture logiche formali della fisica ereditate da Galileo, Newton e Einstein); è stato possibile risolvere solo alcune tipologie di problemi. Negli ultimi anni del secolo appena passato si sono tentate nuove vie. La crescita di una pianta, la descrizione del corso di un fiume, previsioni complesse, realizzazioni grafiche originali, sono tutte questioni che oggi hanno delle precise risposte e sono adeguatamente descritte da nuovi metodi; e che quindi aggiungono un piccolo contributo allo scibile umano. L'apporto sempre più significativo del computer ha permesso di esplorare soluzioni che gradualmente migravano da manipolazioni algebriche a calcolo numerico e algoritmico. È evidente che il programmatore ricopre oggi un ruolo sempre più importante nell'ambito del progresso tecnologico. Anche l'approccio ai problemi è cambiato: la presenza di una entità che faccia calcoli complicatissimi a velocità inimmaginabili, il computer, ha innescato nuovi metodi di soluzione e ha aperto la porta al trattamento di questioni considerate in passato "non trattabili". I sistemi caotici che classificheremo e studieremo tra queste pagine sono un esempio della casistica di problemi appena citati. La presente trattazione ha lo scopo di "fotografare" la situazione attuale, tentando una classificazione circa i sistemi caotici e di approfondire uno soltanto uno dei metodi esaminati. Spesso ricorrerà il termine *frattale* il quale costituisce soltanto un sottoinsieme di ciò che affronteremo. Esordiremo, appunto, con le classiche funzioni frattali, poi vedremo come alcune leggi conosciute della matematica possono produrre caos, esplorando la serie di Newton. Esploreremo, ancora, un importante metodo solo accennato nello scorso appuntamento e che invece sarà approfondito nel prossimo, si tratta degli L-system. Infine, un metodo conosciuto come: "strani attrattori" sarà maggiormente analizzato e per esso si produrrà un semplice programma per testarne le potenzialità.

CLASSICI FRATTALI

Ricordo con soddisfazione come l'argomento frattali, trattato ormai più di tre anni fa, avesse suscitato un notevole interesse. Sicuramente furono apprezzate le importanti applicazioni, ma si rilevò determinante il piacevole effetto grafico che tali funzioni producono. Il padre dei frattali, Benoit Mandelbrot, ha descritto il frattale come: "una forma geometrica frammentata e ruvida che può essere divisa in parti, ognuna di esse (a meno di approssimazioni) è una copia ridotta della stessa". Una definizione puramente matematica delinea un frattale come: "Un insieme di punti la cui dimensione frattale supera quella topologica". Nelle due definizioni si scorgono gli elementi fondanti dei frattali, ovverosia il concetto di autosimilarià e la dimensione frazionaria.
Fractus è una parola latina che significa frastagliato. La traccia descritta da B. Mandelbrot, scienziato dell'IBM nonché Nobel per la matematica, sulla base di una idea di Gaston Julia, prende in esame una funzione complessa (rappresentabile nel piano complesso di Gauss). La funzione è del tipo:

$$zn+1=f(zn)$$

Si tratta di una funzione iterativa per la quale il risultato ottenuto ad un generico passo sarà dato in pasto alla stessa funzione (sarà argomento della funzione) per generare il risultato del passo successivo. In particolare la funzione pensata era:

$$f(zn)=zn\ 2 + z0$$

Si ottiene una serie per ogni valore iniziale di $z0$ descritta sul piano complesso. Mandelbrot focalizzò l'attenzione sul comportamento della serie al varia-

re del punto iniziale z_0. A tale proposito egli classificò quattro possibili comportamenti della serie: nel primo decade a zero; nel secondo tende a infinito; nel terzo oscilla su un numero finito di stati; nell'ultimo caso i valori pur non tendendo a infinito fuoriescono da un fissato pattern. Ogni comportamento viene rappresentato diversamente in modo grafico, si ottiene così il frattale di Mandelbrot. Il pattern è il cerchio di raggio *2*. Quindi la serie esce dal pattern se $|z_0|>2$.
Il semplice programma pascal riportato di seguito costruisce il frattale descritto. Per ragioni di chiarezza, il punto iniziale è identificato con la variabile *c* nelle sue due componenti reali e complesse, c_1 e c_2.

```
// Listato di Mandelbrot
...//dichiarazione di variabili ...
Begin
  ... // Inizializzazione grafica ...

  for i:=1 to 300 do
    for j:=1 to 150 do
      Begin
        c1:=-2+4*i/300;
        c2:=2-4*j/300;
        x:=c1;
        y:=c2;
        k:=1;
        pbreak := false;
        while (k<=30) and (not pbreak) do
          Begin
            k:=k+1;
            xs:=x*x-y*y+c1;
            ys:=2*x*y+c2;
            m:=xs*xs+ys*ys;
            if m>4 then pbreak:=true;
            x:=xs;
            y:=ys;
          End;
        if pbreak then
        Begin
          putpixel(i,j,k);
          putpixel(i,300-j,k);
        End
      End;
      ...
End.
```

Analizzando il codice si può constatare come non ci siano input, infatti: si pone $z=0$ e i valori di *c* (z_0) vengono fatti variare come era stato descritto. c_1 e c_2 si ottengono da una trasformazione dei valori che cambiano (soggetti a iterazione), ovvero *i* e *j*, la divisione per *300* consente di restringere il range *1..300* a *0..1*. L'insieme costruito è a colori, per farlo si è usata la variabile *k* che ci indica a quale iterazione la funzione esce dal cerchio di raggio *2*. Si ricorda che, nella modalità grafica utilizzata, i colori variano tra *1* e *32*. L'output prodotto dal programma è riportato in **Figura 1**.
Osservando la figura notiamo come la zona nera, ovvero i punti interni alla figura prodotta, siano il risultato del primo degli eventi classificati da Mandelbrot, ossia il decadimento a zero della serie. La porzione esterna (che cambia colore fino a diventare verde) è generata dalla tendenza a ingrandirsi (che successivamente, con il verde, porta verso valori che divergono a infinito).
Come è facile intuire, queste tecniche sono state il punto di partenza di una quantità molto consistente di altri studi, che una semplice enunciazione richiederebbe lo spazio di più di una rivista. A titolo di esempio, l'identica tecnica applicata ad una differente funzione genera altri interessanti sistemi caotici. In particolare, la funzione di C.A. Pickover, successivamente battezzata "biomorfe", è la seguente:

Fig. 1: **Frattale classico di Mandelbrot**

$$f(z_n)=sen(z_n)+ ez + c$$

Essa è usata in applicazioni biologiche che sono distinguibili assegnando diversi valori di c. In **Figura 2** due si possono osservare i diversi output in funzione della variabile *c*.

Fig. 2: **Forme frattali biomorfe**

SERIE DI NEWTON

Newton è stato citato come il classico, ma ironia della sorte, i suoi studi sono ancora il punto di partenza per quello che potremmo considerare un metodo alternativo per l'analisi di alcuni sistemi caotici. La serie di Newton per la ricerca di radici di una equazione polinomiale, assume la forma:

$$f(z) = a_0 + a_1 z + a_2 z^2 + \ldots + a_m z^m = 0$$

Il metodo genera una serie, dove la *n+1esima* approssimazione alla soluzione è data da:

$$z_{n+1} = z_n - f(z_n) / f'(z_n)$$

$f'(z_n)$ è la pendenza (derivata) di $f(z)$ valutata su z_n.
Per creare un'immagine 2D, usando questa tecnica, ogni punto che si trova in una partizione del piano indica se colorare o meno il punto di partenza z_0, a seconda se soddisfa alcune condizioni. Cosicché, la soluzione è proprio z_0. Stabilito un target, una solu-

AUTOSIMILARITÀ

Due figure sono simili se hanno la stessa forma. Quando questi caratteri di "somiglianza" sono in un'unica figura, cioè una parte di essa è simile ad un'altra parte più grande, allora diremo che tale figura gode della proprietà di autosimilarità. È il caso del triangolo di Sierpinski o del fiocco di neve di Van Kock.

zione che si intende ottenere, ad esempio l'appartenenza del risultato ad un insieme (nel caso più semplice l'appartenenza di un punto ad una circonferenza), la colorazione del punto dipende dal raggiungimento o meno della soluzione, oppure dal numero di passi che sono occorsi per arrivarci. Questo secondo parametro può essere distinto con colori diversi. Ad esempio, si colora il punto (corrispondente a *z0*) di nero se si arriva subito alla soluzione. Con colori che tendono al bianco si rappresentano soluzioni raggiunte con un maggiore numero di iterazioni. Un semplice esempio è un'applicazione della serie di Newton per la funzione polinomiale *z*z*z-1*, per la ricerca delle tre radici. In **Figura 3** viene mostrato il sistema caotico rappresentato graficamente. Si deduce come solo piccole porzioni di piano complesso raggiungano subito la soluzione. Una caratteristica significativa dei sistemi caotici è che, pur mantenendo uguali tutte le condizioni, una piccola variazione dello stato iniziale può provocare risultati completamente differenti. In **Figura 3** si può osservare come alcuni punti iniziali facciano convergere rapidamente il polinomio verso la soluzione (zona nera), mentre altri convergono molto più lentamente.

Fig. 3: ***Sistema caotico generato dalla serie di Newton applicata a z3-1=0***

L-SYSTEM

I sistemi di *Lindermayer* o semplicemente *lsystem*, sono tra le applicazioni più interessanti nell'ambito che stiamo esplorando. Si tratta di definire una coppia di regole chiamate assioma e generazione (o analogamente produzione). Propongo una definizione molto generale. Prendendo un oggetto, questo viene sostituito da un altro, individuando su esso le istanze che descrivono l'assioma e cambiandole con la produzione.
Il tutto si basa su questa semplice quanto potente attività di sostituzione di assiomi con produzioni. Il discorso è più chiaro se il procedimento si applica a un set di stringhe. Esisterà una stringa iniziale, l'assioma che rappresenta il punto di partenza, questa viene sostituita in ogni suo simbolo secondo le regole descritte dalla produzione. Il procedimento può essere iterato: ad ogni passo, si può sottoporre il risultato ottenuto nuovamente alle regole di generazione definendo così nuove stringhe. Facciamo un esempio. Supponiamo di avere come assioma la stringa: *F+F* e come produzione la regola *F--> F+FF-F+F.* Applicare il metodo significa sostituire ogni simbolo dell'assioma con la produzione; si ottiene così:

F+FF-F+F+F+FF-F+F

Come dicevamo possiamo iterare il procedimento. Non si deve far altro che applicare la stessa regola di produzione alla stringa appena ottenuta. Si genera il risultato della iterazione 2. Eccolo:

F+FF-F+F+F+FF-F+FF+FF-F+F-F+FF-F+F+F+FF-F+F+F+FF-F+F+F+FF-F+FF+FF-F+F-F+FF-F+F+F+FF-F+F

Già alla seconda delle iterazioni la cosa si complica, ovviamente si ottengono risultati sempre più "corposi" con l'aumentare delle iterazioni. Il fattore di crescita dipende verosimilmente dal tipo di produzione che si attua. È stato provato come sistemi caotici di questo tipo siano adatti alla descrizione della crescita di piante. In **Figura 4** è riportato un esempio.
Una delle più importanti caratteristiche degli *lsystem* è che hanno bisogno di poche informazioni per essere rappresentati. La trasposizione grafica sulla base del risultato ottenuto si ottiene applicando specifiche regole. Ma queste sono attività che ci riserviamo per la prossima puntata.
Il triangolo di Sierpinski e il fiocco di neve di Van Kock possono essere prodotti attraverso *lsystem* andando a definire opportunamente gli assiomi e le produzioni. Chi volesse approfondire questo ultimo concetto può fare riferimento all'articolo della sezione soluzioni dello scorso numero.

Fig. 4: ***lsystem applicato alla crescita di una pianta***

GLI STRANI ATTRATTORI

Un altro sorprendente metodo usato per descrivere sistemi caotici sfrutta coppie di espressioni matematiche iterative per le quali i risultati saranno input delle stesse espressioni ai passi successiie ed in modo incrociato. Mi spiego meglio con un esempio. Consideriamo la coppia di equazioni:

xn+1 = yn – sign(xn)|bxn-c|0.5
yn+1 = a - xn

Si può notare come *x* sia un input per *y* e viceversa. I tre valori identificati da *a, b* e *c* sono delle costanti. La rappresentazione grafica di questi punti (coppia di coordinate) mostra strani attrattori. Un'analisi minuziosa svela come i punti saltino ad ogni iterazione all'interno del piano. Il programma di seguito costruisce uno strano attrattore per le equazioni riportate sopra. Per chiarezza è riportato nella sua integrità. È divertente vedere le figure generate cambiando i valori delle tre costanti, nonché il punto iniziale.

```
program Julia1;
uses graph;
type punto=record
        x,y:real;
        end;
var det,g : integer; {variabili di configurazione grafica}
          i : integer;
xs,ys,col,k : integer;   { coordinate e colori di schermo }
      a,b,c,d : real; { costanti iniziali }
      pn,pv : punto;  {  punto nuovo e vecchio}
function strange1(x,y:real) :punto;
var pres:punto;
    xn,yn:real;
    signx:integer;
begin
     if x<0 then signx:=-1
     else signx:=1;
     xn:=y-signx*sqrt(abs(b*x-c));
     yn:=a-x;
     pres.x:=xn;
     pres.y:=yn;
     strange1:=pres;
end;

begin
   { input delle costanti e valori iniziali }
   writeln;
   writeln('inserisci le tre costanti');
   write(' a  --> ');
   readln(a);
   write(' b --> ');
   readln(b);
   writeln;
   write(' c --> ');
   readln(c);
   writeln('inserisci il punto iniziale');
   write(' x  --> ');
   readln(pv.x);
   write(' y --> ');
   readln(pv.y);
   det:=detect;
   initgraph(det,g,'');
   { Segnalazione di inizio ciclo }
   setcolor(green);
   circle(695,100,10);
   k:=1;
   for i:=0 to 20000 do
      Begin
        pn:=strange1(pv.x,pv.y);
        xs:=300+k*round(pn.x);
        ys:=300-k*round(pn.y);
        col:=(i div 5000)+1;
        putpixel(xs,ys,col);
        pv:=pn;
      End;
   { Segnalazione di fine ciclo }
   setcolor(red);
   circle(700,100,10);
   readln;
   closegraph;
end.
```

La prima fase prevede gli input delle tre costanti e dei punti iniziali della serie. Poi, un ciclo di 20000 iterazioni calcola le coppie di coordinate. *pn* è il nuovo punto (record con i due campi rappresentanti ascissa e ordinata), *pv* è il punto calcolato alla iterazione precedente che serve come input dell'espressione corrente. Le due variabili di schermo *xs* e *ys* sono opportunamente modificate per essere adeguatamente raffigurate in forma grafica. L'output generato per diverse prove è rappresentato in **Figura 5**. I valori riportati nella **Tabella 1** sono gli input del programma.

Fig. 5: ***Output per diverse esecuzioni del programma di generazione di strani attrattori. Si consulti la Tabella 1 per visionare gli input corrispondenti***

grandezza\figura	A	B	C	D1	D2	E	F
a	0.22	-1.1	0.2	-0.5	-0.5	23.5	23.5
b	-40	-2.1	-40	2	2	-2.3	-2.3
c	200	-400	200	10	10	-350	-350
x_0	0	0	0	0	20	0	0
y_0	0	0	0	0	3	0	2

Tabella 1: I valori delle costanti relativi alle varie figure

CONCLUSIONI

Il viaggio si è concluso. Abbiamo dato uno sguardo ad alcune tecniche oggi usate per la implementazione di sistemi caotici e abbiamo approfondito alcuni di essi con la produzione di semplici programmi per la rappresentazione grafica. La prossima volta visioneremo altri metodi e ci soffermeremo sugli lsystem oggi solo accennati. Vi aspetto per la prossima puntata.

Fabio Grimaldi

Quadrati magici, tris e altri giochi su griglie quadrate

La zuppa di pesce

Una divertente variante del gioco del quindici è "fish soup". Questa volta i due sfidanti per prevalere nella competizione, devono scegliere parole invece di numeri.

I TUOI APPUNTI

Utilizza questo spazio per le tue annotazioni

REQUISITI

Conoscenze richieste
Nozioni di logica e aritmetica

Software
Nessuno

Impegno

Tempo di realizzazione

Per soddisfare i palati più esigenti, menu raffinato a base di pesce, questo mese. Dovremo però stare attenti a scegliere i giusti ingredienti per evitare un indigesto miscuglio. Come consuetudine, chiuderemo la pagina aperta nello scorso appuntamento, dando le tecniche di soluzione del gioco del 15. Dopo aver esaminato il nuovo gioco tra queste pagine, individueremo un percorso risolutivo. Infine, accenneremo i sorprendenti quadrati magici, che saranno il piatto forte della prossima puntata.

SOLUZIONE PER IL GIOCO DEL 15

Il gioco è talmente semplice nella formulazione che rivisitare le regole ci costerà poche righe. Due sfidanti a turno devono scegliere un numero (che non può essere considerato più di una volta) compreso nell'intervallo [1, 9]. Come obiettivo, sommando i propri numeri a disposizione si deve totalizzare 15. Naturalmente, vince chi prima arriva al risultato. Non so quale sia la strategia vincente che voi lettori abbiate approntato, la mia prevede la fusione di altri due famosi giochi (o rompicapo, come preferite), i conosciuti: tris (o tic tac toe) e quadrati magici. Ma procediamo per gradi, vedremo come costruendo la strategia risolutiva magicamente appariranno gli enigmi. Se nel giocare non schematizziamo in una struttura adeguata i numeri, difficilmente potremo avere la meglio sul nostro avversario, anzi, qualora egli conosca a fondo il gioco rischieremo di perdere. Faccio un esempio per far capire la necessità di possedere una strategia. Supponiamo che i soliti giocatori Tizio e Caio si affrontino ancora. Questa volta, però, Tizio ha precedentemente preso lezioni di strategia. Supponendo che tocchi al nuovo stratega scegliere, esordirà optando per il 5.
Un percorso plausibile per Caio, che comunque è sprovvisto di nozioni teoriche, potrebbe essere scegliere un numero grande che possa garantire con due sole prese la vittoria, quindi sceglie 9. Tizio risponde con 6. Con questa mossa ha praticamente vinto, poiché ha parato la minaccia di Caio di prevalere al prossimo turno e si è garantito una doppia possibilità di chiudere la partita nell'immediato futuro. Caio è costretto a impedire la vittoria immediata di Tizio scegliendo 4 (infatti: 6+5+4=15). Tizio fa la sorprendente scelta 2. A questo punto, alla prossima tornata, potrà vincere con la doppia terna (2,5,8) o (2,6,7); avrà quindi uno dei due numeri 8 e 7 per vincere. Caio perde potendo parare solo una delle due minacce di vittoria. L'esempio è esplicativo. Ma quale è il disegno vincente alla base delle scelte di Tizio? La prima sinapsi, intesa come collegamento tra due cose apparentemente estranee, è quella tra il gioco in questione e i quadrati magici di dimensione 3 (sono approfonditi nell'ultimo paragrafo). Perché? Per entrambi è ricorrente la realizzazione di una somma 15, ed in entrambi i casi si ha a che fare con numeri che vanno da 1 a 9 scelti una volta. Allora, costruiamo un quadrato magico di dimensione 3.

8	1	6
3	5	7
4	9	2

Rileggendo l'esempio di partita prima descritto, tra Tizio e Caio, si svelano tutti gli arcani sulla questione. Per facilitare la cosa, le scelte di Tizio possono essere contrassegnate con delle croci sui numeri selezionati, mentre, quelle di Caio con cerchietti. Ecco allora che è entrato in scena anche l'ultimo elemento: il gioco del tris. Questo ultimo non è nella sua versione originale, poiché portano alla vittoria anche le due coppie 9 e 6 e 8 e 7. La strategia è quindi quella ormai conosciuta del tris, con particolare attenzione per la variante.
La difficoltà incontrata nel definire una tecnica corretta è stata soltanto la costruzione di un'adeguata struttura su cui sistemare i dati e l'individuazione del legame con i giochi del quadrato magico e del tris.

LA VARIANTE SUCCULENTA

La variante più gustosa del gioco del 15 prevede come oggetto delle nostre manipolazioni non sem-

plici numeri ma parole, alcune anche commestibili! Ricordo che si tratta delle seguenti:

VOTE	KNIT	ARMY	CHAT	FISH	SOUP	HORN	SWAN	GIRL

I due contendenti devono a turno scegliere una delle parole proposte. L'obiettivo è trovare tre parole contenenti la stessa lettera. Ad esempio, la sequenza "army horn girl" è vincente per la presenza tripla della lettera **R**. Risolviamo anche questo enigma. Un'analisi lessicale può aiutarci nel compito. La prima cosa di cui mi sono reso conto quando ho osservato le parole, è che esse sono costituite da due categorie di lettere, che ho distinto in utili e inutili. La F di fish, ad esempio, ci servirà a ben poco, dato che non è presente in alcuna altra parola, senza tanti fronzoli l'ho marcata come inutile (solo in questo contesto, sia ben chiaro). Altre lettere, invece, si presentano esattamente in tre parole, sono quindi utili. I due set sono ben definiti. Le lettere inutili sono: F, U, P, W, G, L, M, E, V, K, T, C; mentre l'insieme delle lettere utili è: I, S, H, O, A, N, R, T. Andando a verificare la presenza degli elementi del secondo set all'interno delle singole parole ho notato con sorpresa che alcune ne contenevano di più di altre. In particolare, quattro (GIRL, SOUP, ARMY e VOTE) contengono solo due lettere utili; un altro gruppo di quattro (FISH, SWAN, KNIT, CHAT) ne include ben tre utili, ed una sola inutile; ed infine, la sola parola HORN è composta nella sua totalità da segni dell'alfabeto utili. Sulla base della appartenenza a questi gruppi, e nel rispetto delle lettere comuni tra le varie parole, si possono ancora una volta incasellare gli elementi in questione in una matrice tipica del gioco del tris. Come accade in questi casi è più facile realizzare la cosa che spiegarla. Ecco come apparirà il nuovo quadrato costruito:

FISH	SOUP	SWAN
GIRL	HORN	ARMY
KNIT	VOTE	CHAT

Questa volta il gioco si è ricondotto al solo tris, per intenderci senza passare per il quadrato magico come nel caso precedente. Le terne vincenti sono otto e sono le tre righe, le tre colonne e le due diagonali. Certo la costruzione è stata più difficile e non è affatto semplice da concepire, richiede infatti, diversi passaggi logici. Una volta pervenuti a tale risultato sarà più semplice giocare, anche perché in questo caso, si tratta del gioco del tris tradizionale, ossia non variato come per il gioco del 15.

PER LA PROSSIMA MAGICA PUNTATA

Apriamo un altro file molto interessante, un magic file. I quadrati magici che introdurremo saranno studiati e approfonditi alla prossima puntata. Un quadrato magico è una serie di numeri, distinti fra loro, disposti su una griglia quadrata (con stesso numero di colonne e righe) che hanno particolari proprietà, considerate tempo fa "magiche". Premesso che **n** è la dimensione del quadrato (matrice), la proprietà fondamentale è che comunque si sommino **n** numeri, per qualsiasi riga, colonna o diagonale (ve ne sono due: la prima va dall'elemento (1,1) all'elemento (n,n) e la seconda va dall'elemento (1,n) a quello (n,1)), tale somma risulti essere sempre lo stesso numero magico. Facciamo l'esempio più semplice, quello di una matrice 3 x 3. In questo caso la soluzione è unica se non si considerano quadrati magici ottenuti dal seguente per rotazione intorno al centro o per ribaltamento. Ecco il primo quadrato magico:

6	1	8
7	5	3
2	9	4

Come di può verificare la somma della prima riga è il numero magico 15, così come la somma dei numeri della seconda e terza riga. Il numero magico 15 si ripresenta anche se si sommano i numeri per le tre colonne e le due diagonali.
Facciamo un altro esempio, visioniamo un quadrato magico più grande, di dimensione 4. Uno possibile, tra gli 880 costruibili, è il seguente.

16	3	2	13
5	10	11	8
9	6	7	12
4	15	14	1

Storia, teoria, analisi e trucchi alla prossima puntata.

CONCLUSIONI

A proposito di zuppa. Se trovate un set di parole in italiano mandate la soluzione al mio indirizzo fabg@edmaster.it, sono molto interessato alla cosa.
Confesso che ho provato per diverse ore senza ottenere il risultato sperato. Ho però individuato delle semplici regole da rispettare per raggiungere l'obiettivo, che ho codificato e ho riportato nel box zuppa all'italiana.
Un eventuale successo troverà sicuramente spazio su queste pagine. Spero che il menu non abbia procurato indigestioni ma al contrario sia stato ben gradito.
Vi aspetto per nuovi enigmi da risolvere!

Fabio Grimaldi

UN ZUPPA ITALIANA

Per costituire un set di parole italiane che siano appropriate per il gioco fish soup, come specificato nell'articolo bisogna rispettare una serie di regole:

1. **Individuare un set di lettere utili, di cardinalità otto.**
2. **Distribuire le lettere utili su nove parole in modo che quattro di esse ne contengano due, altre quattro ne contengano tre e una le includa tutte e quattro;**

Aiutarsi costruendo la griglia tipica del tris. È consigliabile che le vocali si dividano tra il set delle inutili e quello delle utili. A rigore le parole possono non essere di lunghezza quattro (ma maggiore), e le lettere inutili presentarsi non una volta (anche due), funziona ugualmente. Certo, se anche queste ultime due condizioni sono verificate si ottiene una zuppa di pesce pura !!

Connettersi al Web via Http con Jakarta Commons HttpClient

Il protocollo HTTP è uno dei protocolli fondamentali su cui si basa il funzionamento della rete Internet. Tramite HTTP è possibile infatti connettersi ad un sito Web, scaricare risorse come file HTML, immagini o documenti oppure anche fornire dei dati ad una pagina Web per ottenere in risposta un elenco dinamico. Nello sviluppo di applicazioni Java può risultare necessario eseguire una di queste operazioni specialmente in applicazioni come:

- software che eseguono test funzionali su siti Web, per verificarne il corretto funzionamento
- software che eseguono stress test su siti Web, allo scopo di misurare come il server si comporta in condizioni di carico (numero di utenti connessi in contemporanea) definite

Sebbene il package java.net offra alcune funzionalità di base per l'accesso a risorse via HTTP, si verifica che spesso queste sono insufficienti all'atto pratico. Ci viene in contro un'ottima libreria prodotta dall'Apache Group denominata *Jakarta Commons HttpClient*, che rende possibile esprimere con API Java tutte le più comuni operazioni che eseguiamo durante la navigazione in un sito Web.

Filippo Diotallevi

\CODICE\ JakartaCommonsHttpClient.zip
\LIBRERIE_JAVA\commons-httpclient-2.0.2.zip

‹1› LA LIBRERIA HTTPCLIENT

La classe principale della libreria è *HttpClient*, che permette di controllare completamente le connessioni HTTP attraverso cui è possibile eseguire tutti i metodi previsti dalla specifica (*GET, POST, PUT, DELETE, HEAD, OPTIONS*, e *TRACE*), comandare l'apertura di una sessione ed anche impostare eventuali Cookie.

‹2› CARICARE UNA RISORSA VIA HTTP

```
public String printMainPage() throws
                    HttpException,IOException {
    HttpClient client = new HttpClient();
    HttpMethod method = new
            GetMethod("http://www.google.it");
    client.executeMethod(method);
    return method.getResponseBodyAsString();}
```

Volendo scaricare una pagina da un sito, possiamo utilizzare il metodo *GET* che ha il suo corrispondente nella classe *GetMethod* della libreria. Il codice necessario per connettersi a *www.google.it* e scaricarne la home page è quello mostrato in questo passo. È bene considerare le eccezioni che si possono presentare collegandosi a Interne, in caso contrario basterebbe una semplice assenza di connessione per generare un errore ed un blocco dell'applicazione.

‹3› LA CLASSE DA RICHIAMARE

```
org.apache.commons.httpclient.methods
Class GetMethod

java.lang.Object
  └org.apache.commons.httpclient.HttpMethodBase
      └org.apache.commons.httpclient.methods.GetMethod
```

L'API è semplice ed intuitiva. *HttpClient* fa da intermediario tra l'applicazione ed il server remoto, mentre le modalità di connessione sono specificate attraverso una classe che implementa l'interfaccia *HttpMethod*. Nel nostro caso è *GetMethod*, ma esistono altre implementazioni che corrispondono ad altri metodi della specifica HTTP (*PostMethod, PutMethod, DeleteMethod*...).

‹4› INTERPRETARE LE RISPOSTE DEL METODO

SC_OK
200 OK (HTTP/1.0 - RFC 1945)

SC_PARTIAL_CONTENT
206 Partial Content (HTTP/1.1 - RFC 2616)

SC_PAYMENT_REQUIRED
402 Payment Required (HTTP/1.1 - RFC 2616)

Il metodo *executeMethod* di *HttpClient* restituisce lo stato finale della risposta. È una buona idea verificare sempre il suo valore per controllare che contenga codici corretti (il codice tipico di *OK* è il 200, per conoscere il significato degli altri codici basta confrontarli con le costanti esposte dalla classe *HttpStatus*).

‹5› COME GOOGLE SI INTERFACCIA AL SUO DATABASE

Per ottenere il contenuto della risorsa richiesta si utilizza il metodo *getResponseBodyAsString* della classe *HttpClient*, oppure *getResponseBody* che restituisce la risorsa richiesta sotto forma di array di byte. Osserviamo che l'URL della pagina visualizzata quando si cerca "ioprogrammo" è del tipo *http://www.google.it/search?q=ioprogrammo*. Il parametro *q* contiene la parola cercata.

‹6› INTERROGHIAMO IL DATABASE DI GOOGLE

```
public String printIoProgrammoPage() throws
                    HttpException, IOException
{
    HttpClient client = new HttpClient();
    HttpMethod method = new
        GetMethod("http://www.google.it/search");
    //imposto la query
    NameValuePair[] params = new NameValuePair[1];
    params[0] = new NameValuePair("q","ioprogrammo");
    method.setQueryString(params);
    //imposto lo User-Agent
    method.addRequestHeader(new Header(
                "User-Agent","Firefox 0.9.2"));
    client.executeMethod(method);
    return method.getResponseBodyAsString();
}
```

Costruiamo la stringa di ricerca, imitando il comportamento di Google: passiamo il parametro *q* con la chiave di ricerca e aggiungiamo un header alla richiesta HTTP, simulando il funzionamento di un browser reale. Analizzando la risposta, si può verificare che effettivamente è stata caricata la pagina dei risultati attesa.

Gestire l'upload di documenti nelle pagine Jsp

Cresciuti in numero e qualità nel corso degli anni i Jakarta Commons sono un insieme di librerie Java che vengono incontro allo sviluppatore risolvendo, in modo rapido ed efficace, un certo numero di problemi tipici della programmazione. In questo articolo tratteremo l'utilizzo di *FileUpload*. *Jakarta FileUpload* fornisce al programmatore un set di classi Java che permette all'utente di una applicazione Web di caricare dei file sul server. Come esempio, si pensi ai siti di ricerca di personale che permettono agli iscritti di caricare il proprio curriculum, o applicazioni WebMail che permettono di inserire allegati. L'applicazione che andremo a sviluppare consente all'utente di selezionare due file dalla propria macchina e, alla pressione del tasto di conferma, di effettuare l'upload di tali file sul server salvati nella cartella del Servlet Container che contiene il codice da noi sviluppato.

Filippo Diotallevi

<1> IL FILE HTML PER L'INTERFACCIA UTENTE

```
<HTML>
<HEAD><TITLE>Demo di Jakarta
                    FileUpload</TITLE></HEAD>
<BODY>
<h2>Demo di Jakarta FileUpload</h2><br/><br/>
<FORM name="files" action="FileUpload.jsp"
    method="post" enctype="multipart/form-data">
<table width="60%" cellspacing="3"
                  cellpadding="3" border="0">
<tr>
 <td>
  Primo file:</td><td><input type="file"
                               name="file1"/>
 </td></tr>
<tr>
 <td>
  Secondo file:</td><td><input type="file"
                               name="file2"/>
 </td></tr>
<tr>
 <td> </td>
 <td>
  <input type="submit" name="Submit"
                      value="Carica i file"/>
 </td>
</tr>
</table></FORM>
</BODY></HTML>
```

Il primo passo per costruire l'applicazione consiste nel creare una semplice pagina HTML che permetta all'utente di selezionare due file presenti sul proprio Pc. Come si può vedere dal listato, si tratta di una comune pagina HTML con all'interno un form.I tratti distintivi di questo form sono l'encoding type che è di tipo "multipart/form-data" e i campi di input che sono di tipo "file".
Queste indicazioni sono utilizzate, dal server http, per interpretare correttamente le caratteristiche del form e per presentare la pagina (che nel nostro caso chiameremo *index.html*) così come appare in Figura.

<2> L'IMMISSIONE DEL PERCORSO DEL FILE

Il form, come indicato dall'attributo *action*, alla pressione del pulsante di sottomissione, si fa carico di inviare una richiesta HTTP alla pagina JSP denominata *FileUpload.jsp*. La scelta di far gestire l'upload dei file ad una pagine JSP è fatta per praticità, in quanto il nostro esempio risulta particolarmente semplice. In scenari più complessi, sarà meglio gestire queste operazioni attraverso una opportuna Servlet con tutti i controlli del caso: sicurezza e prestazioni saranno sicuramente maggiori.
Nel caso utilizzassimo una servlet, limitatamente alla gestione dei file, logica applicativa e codice non muterebbero in modo significativo.

<3> IL CODICE CHE EFFETTUA L'UPLOAD DEL FILE

```
DiskFileUpload fu = new DiskFileUpload();
List fileDaCaricare = fu.parseRequest(request);
Iterator listaFile = fileDaCaricare.iterator();
```

In *FileUpload.jsp* entra in gioco *Jakarta FileUpload*. La prima classe da utilizzare è *DiskFileUpload*, che ci permette di estrarre dalla request HTTP corrente la lista dei *FileItem* selezionati dall'utente.
Una volta ottenuta la lista dei file è sufficiente iterare su ogni elemento di questa lista, facendo attenzione a controllare che si tratti di un campo del form (tramite la verifica *isFormField()*).

<4> CONCLUDIAMO L'OPERAZIONE

```
while(listaFile.hasNext()) {
 //considera un File per volta
 FileItem file = (FileItem)listaFile.next();
 //Considera solo i form field
 if(!file.isFormField()) {
  //Internet Explorer restituisce il path assoluto
            mentre gli altri browser quello relativo
  String fileName=file.getName();
  int posizioneSeparator =
            fileName.lastIndexOf(File.separator);
  if (posizioneSeparator != -1) //se esiste un separatore
   fileName = fileName.substring(
                        posizioneSeparator+1);
              //estrae il nome del file dal path
  File nuovoFile= new File(application.getRealPath(
               "/"), fileName);  //salva il file
  System.out.println("Sto salvando il file" +
                   nuovoFile.getAbsolutePath());
  file.write(nuovoFile);
 }
}
```

Al fine di uniformare il comportamento dell'applicazione al variare del browser viene controllata la presenza del carattere di separazione ('/' oppure '\') nel nome del file, ed in caso affermativo dal path assoluto si estrae il nome del file. Infine, ogni file viene salvato sul server utilizzando il metodo *write* della classe *FileItem*. Per poter provare l'applicazione è ora necessario inserire il listato presentato qui sopra nel corpo della Jsp *FileUpload.jsp*, ed effettuare il deploy di questo file e dell'*index.html* su un Servlet Container come Jakarta Tomcat.

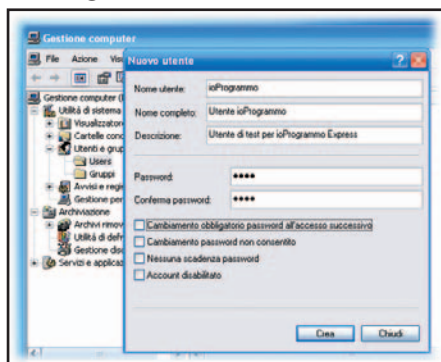
PER SAPERNE DI PIÙ

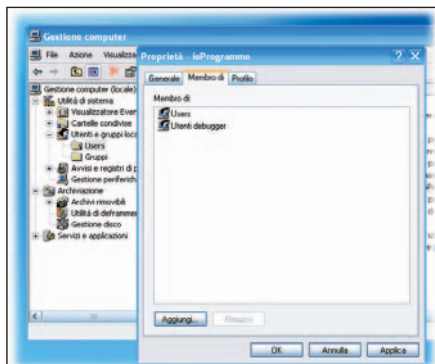
Jakarta FileUpload è una libreria open source, liberamente scaricabile e distribuibile sviluppata all'interno del progetto Apache Jakarta. Maggiori informazioni e modalità di download sono presentate al sito http://jakarta.apache.org/commons/fileupload

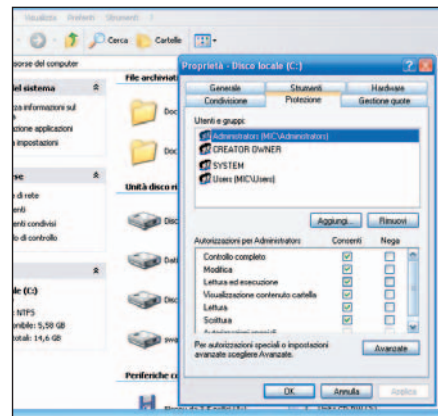
Aumentiamo la sicurezza riducendo i privilegi

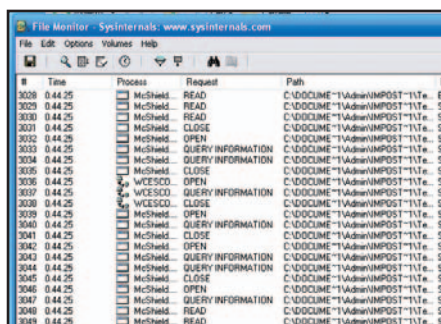
La sicurezza è un tema tanto importante quanto, troppo spesso, trascurato. Siamo abituati a sentir parlare di nuovi virus, worm o attacchi informatici quasi quotidianamente.
Proteggiamo i nostri dati con antivirus sempre aggiornati (spero), firewall ed altri sistemi di protezione, eppure commettiamo spesso un errore abbastanza grave: usiamo il pc con i privilegi di Amministratore.
Come sappiamo, la maggior parte dei sistemi operativi gestiscono svariate tipologie di utenti e di permessi. Ogni tipologia di utente (*user, guest, administrator* ecc .) ha permessi (privilegi) più o meno elevati per eseguire determinate operazioni sul sistema operativo. Si va dalla semplice lettura dei file, ad operazioni di configurazione più o meno complesse.
I sistemi operativi di casa Microsoft ci hanno abituato ad essere Amministratori. Di default, l'utente creato in fase di installazione del sistema rientra nel gruppo degli *Administrators*, quindi, ha i massimi privilegi. Tale condizione è corretta in quanto, senza privilegi elevati, non potremmo installare quasi nulla. Allo stesso tempo, però, molti degli attacchi sopra citati sfruttano i nostri stessi privilegi per compiere azioni indesiderate.
Vedremo, in questo breve tutorial, come creare un nuovo utente con bassi privilegi ed utilizzarlo per le normali operazioni, compreso lo sviluppo ed il debug delle applicazioni.

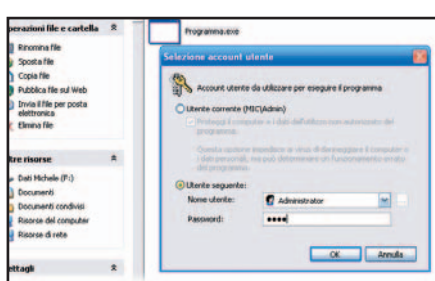
Michele Locuratolo

‹1› CREAZIONE DI UN NUOVO UTENTE

Dal pannello di controllo selezioniamo la voce *"Strumenti di amministrazione"* / *"Gestione Computer"* / *"Utenti e gruppi locali"*. Clicchiamo con il tasto destro del mouse su *"Users"* e selezioniamo *"Nuovo Utente"*. Compilati tutti i campi come in figura, clicchiamo su *Crea*. Il nuovo utente apparirà nella lista insieme con gli altri.

‹2› LA GESTIONE DEI GRUPPI

Selezioniamo il nuovo utente appena creato ed accediamo ai dettagli con un doppio click. Nella scheda *"Membro di"* assicuriamoci che non sia presente la voce *"Administrators"*. Se c'è, eliminiamola. Clicchiamo poi su *"Aggiungi"/ "Avanzate" / "trova"* e dall'elenco selezioniamo *"Utenti debugger"*.
Confermiamo tutte le maschere fino a tornare all'elenco degli utenti.

‹3› LA GESTIONE DEI PERMESSI

Il nostro utente è stato creato, ed appartiene al gruppo degli *User*. Assicuriamoci, ora, che abbia i permessi di accesso alle directory ed ai file presenti nei nostri Hard Disk. Per farlo, clicchiamo con il tasto destro del mouse sull'HD, selezioniamo la voce *"Proprietà"* e la scheda *"Protezione"*. Se il gruppo *Users* dovesse mancare, aggiungiamolo.

‹4› RISOLVIAMO I PROBLEMI

Uno dei problemi che più spesso incontreremo è relativo all'assenza di permessi per la scrittura di alcuni file. Per capire quale file il programma stà cercando di scrivere ed in quale directory si trova, possiamo utilizzare un tool scaricabile da *www.sysinternals-.com*: *Filemon*.

‹5› L'UTILITY RUNAS

Quando abbiamo la necessità di eseguire programmi che richiedono privilegi elevati, possiamo ricorrere all'utility *RunAs*. Clicchiamo con il tasto destro del mouse sul programma da avviare e selezioniamo la voce *"Esegui come…"*. Dal box che apparirà, selezioniamo *"Utente seguente"* e digitiamo la password corretta.

‹6› NEI NOSTRI SOFTWARE

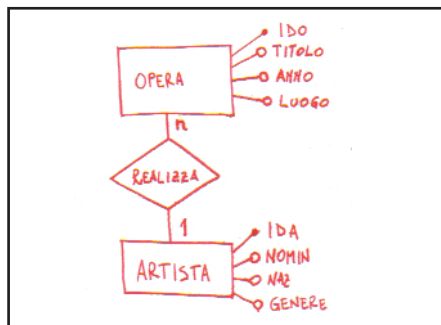
```
private void ScriviLog(string Messaggio){
  StreamWriter sw = new StreamWriter(
    Environment.CurrentDirectory+"\\log.txt"., true);
  sw.WriteLine("["+System
        .DateTime.Now.ToString()+"] " + Messaggio);
  sw.Close();
}
```

Nella scrittura dei nostri software, dobbiamo tenere presente che questi dovranno funzionare anche con permessi limitati. Per evitare spiacevoli inconvenienti, facciamo scrivere i log nella stessa cartella in cui è in esecuzione il programma. Nell'esempio uno stralcio di codice C# che usa *Environment.CurrentDirectory* per recuperarla.

Relazionare tabelle di dati mediante Microsoft Access

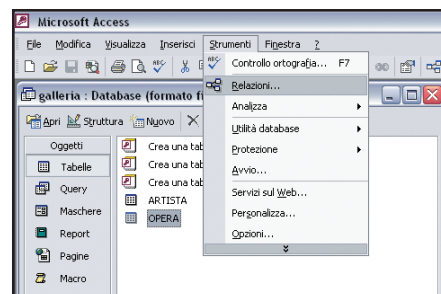
Individuare relazioni tra i dati si può dire che sia la seconda attività da svolgere quando si costruiscono database, dopo aver raggruppato informazioni per entità omogenee, solitamente organizzate in tabelle. Soltanto un uso superficiale, e quindi poco efficiente di un generico DBMS, nel caso specifico Access, non prevede l'associazione dei dati. Capire come si possano trovare le giuste relazioni, oltre a risolvere il problema della ridondanza, rende più chiaro il modello globale della struttura logica dei dati. In MS Access possiamo avvalerci dello strumento *relazioni*. Nell'esempio riportato di seguito ci poniamo nel particolare caso delle relazioni uno a molti (1:n): tale tipologia di associazione prevede che un elemento della prima tabella si relazioni con più elementi della seconda, viceversa, un record della seconda tabella può essere associato ad un unico record della prima. Aggiungeremo nella tabella con relazione "uno" la chiave della tabella "molti", che assumerà il significato di chiave esterna. Ricordo che la chiave è un attributo o un insieme di attributi che identificano, in modo univoco, una istanza di una tabella (con linguaggio formale, una tupla di una relazione).

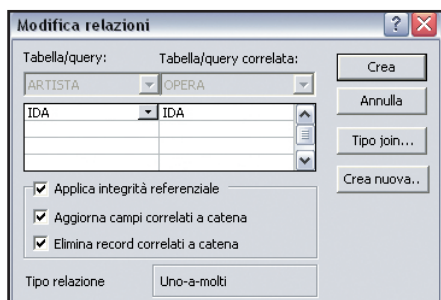
Fabio Grimaldi

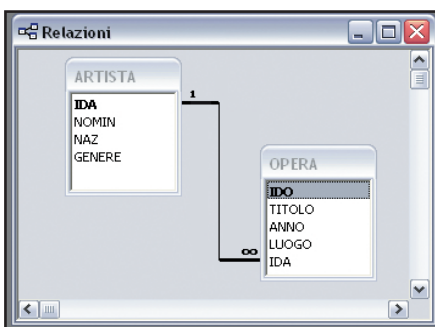
<1> IL MODELLO CONCETTUALE DEI DATI

L'esempio di cui ci serviremo per realizzare le nostre prove è un semplice DB per l'archiviazione di informazioni riguardanti opere d'arte e i rispettivi artisti. In figura è riportato il modello entità/relazione del DB costruito, in cui si possono riconoscere le due entità caratterizzate dai rispettivi attributi, nonché la relazione tra esse.

<2> AGGIUNTA CHIAVE ESTERNA NELLA TABELLA OPERA NEL DB GALLERIA

La relazione uno a molti chiamata *"realizza"*, viene derivata nel modello logico, che successivamente è tradotto in tabelle Access, aggiungendo la chiave dell'entità *molti* (*ARTISTA*), *IDA*, all'interno della entità *uno* (*OPERA*). In Access, una volta costruite le due tabelle, dalla modalità visualizza struttura, bisogna aggiungere alla tabella *OPERA* l'attributo *IDA*.

<3> RICHIAMO E USO DELLO STRUMENTO RELAZIONI

Passo successivo è quello di richiamare, dal menu *strumenti* la voce *relazioni*. Nella finestra che appare possiamo indicare quali tabelle associare. Nel nostro semplice esempio, selezioniamo entrambe le tabelle dalla finestra di dialogo che appare, tenendo premuto il tasto *CTRL* e clicclando sui nomi di tabella corrispondenti. Un clic su *aggiungi* per il passo successivo.

<4> APPLICHIAMO L'INTEGRITÀ REFERENZIALE PER EVITARE LE ANOMALIE

Nell'ambito del pannello attivato, con un drag&drop, si crea l'associazione puntando l'attributo *IDA* (in una qualsiasi tabella) e trascinandolo su *IDA* dell'altra. Ci viene chiesto (figura) se si vuole innescare l'integrità referenziale, ossia un modo per evitare anomalie di cancellazione e modifica per la relazione *1:n* creata. Confermiamo e proseguiamo.

<5> LA RELAZIONE UNA A MOLTI È COSÌ OTTENUTA

Il risultato del procedimento ci porta al risultato sperato. Si ottiene, infatti, una relazione uno a molti (segnalata con i simboli: 1 e ∞). Si noti l'affinità con il modello entità/relazione costruito al primo passo. L'integrità referenziale attivata al passo precedente ci garantirà circa la cancellazione e la modifica di un elemento.

<6> CANCELLAZIONE DI UNA TUPLA IN PRESENZA DI INTEGRITÀ REFERENZIALE

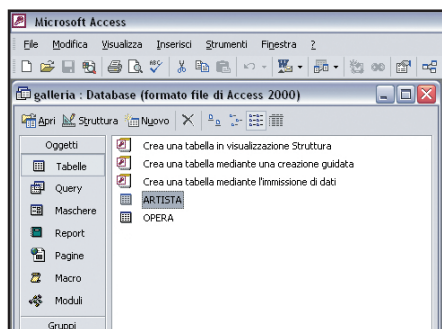
Se ad esempio si vuole cancellare un artista dal DB, a catena verranno eliminati tutti i record correlati nella tabella associata, ossia *OPERA*. Accedendo alla tabella *ARTISTA* e posizionandoci su un artista, ad esempio *Gauguin* con il tasto destro si può eliminare l'intero record. In automatica verranno eliminate tutte le opere dell'artista nella tabella *OPERA*.

Interrogare un DB Access usando lo strumento query

La quantità di dati che può essere contenuta in un database può essere ingestibile se presa nel suo complesso. Tutti i moderni DBMS *(DataBase Management System)* consentono di accedere ai dati attraverso una "proiezione" che limita le informazioni che l'utente deve gestire. Un esempio banale può essere quello dell'ufficio anagrafe: se ad ogni individuo sono associati decine di campi (nome, cognome, codice fiscale, data di nascita, indirizzo, paternità ecc.) e a me interessasse conoscere solo la data di nascita, potrei appunto limitare la visione a soli tre campi del database. Nome, cognome e data di nascita costituirebbero quella che tecnicamente si definisce una vista.

I DBMS mettono a disposizione diversi strumenti per ottenere queste viste, la cui definizione è sempre riconducibile ad una "query SQL", ovvero una descrizione, in SQL, dei dati che vogliamo ottenere. In questa occasione, daremo uno sguardo allo strumento query, messo a disposizione da ACCESS che, per via visuale, consente di costruire query SQL, anche molto complesse, con relativa semplicità. Costruiremo la query a partire da un database in cui sono memorizzate informazioni sulle opere presenti nei più grandi musei del mondo.
Proveremo a effettuare una selezione, per scoprire quali opere sono custodite all'Hermitage di San Pietroburgo.

Fabio Grimaldi

‹1› IL DATABASE DI RIFERIMENTO GALLERIA

Il semplice DB di riferimento *GALLERIA* è costituito dalle due tabelle *OPERA* e ARTISTA che sono messe tra loro in relazione attraverso l'attributo *IDA* (chiave esterna per *OPERA*, e chiave primaria per *ARTISTA*). Tra gli strumenti di Access, oltre alle fondamentali tabelle, si noti la presenza di query.

‹2› UN'OCCHIATA ALLE TABELLE PRESENTI SUL DB

ARTISTA : Tabella

IDA	NOMIN	NAZ	GENERE
1	Monet Claude	Francia	Impressionista

IDO	TITOLO	ANNO	LUOGO
1	Prati a Giverny	1888	San Pietroburgo
2	Il ponte di Waterloo	1903	San Pietroburgo
3	Ninfe blu	1918	New York
9	Impression, soleil levant (l'alba)	1872	Parigi
ıtatore)		0	

IDA	NOMIN	NAZ	GENERE
2	Gauguin Paul	Francia	Impressionista

IDO	TITOLO	ANNO	LUOGO
4	Les parau parau (parole, parole)	1981	San Pietroburgo
5	Arearea (giocosità)	1892	Parigi
ıtatore)		0	

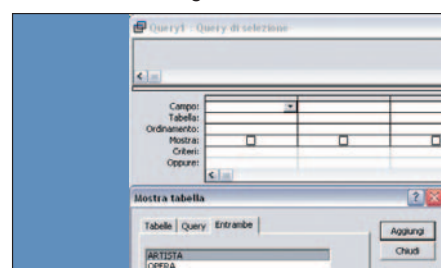
IDA	NOMIN	NAZ	GENERE
3	Matisse Henri	Francia	Espressionista

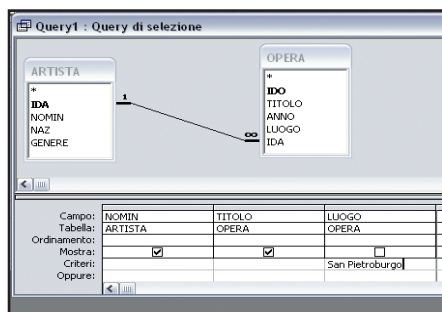
IDO	TITOLO	ANNO	LUOGO
6	Pesci rossi	1912	Mosca
7	La danza	1910	San Pietroburgo
8	Icaro (figura VIII da "Jazz")	1946	Parigi
ıtatore)		0	

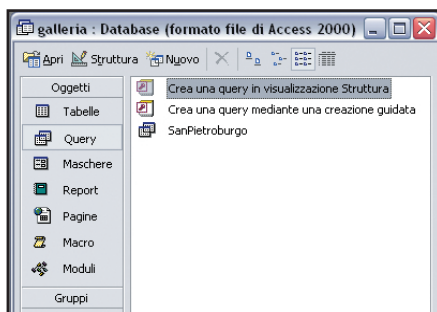
(Contatore)			

Record: 1 di 4

Apriamo la tabella *OPERA* e, per ogni record, esaminiamo i record collegati, cliccando sul tasto +. Si notano, oltre alle informazioni proprie degli artisti, anche le informazioni correlate delle opere archiviate.

‹3› RICHIAMO E USO DELLO STRUMENTO QUERY

Richiamiamo lo strumento *query*. Possiamo realizzarne una attraverso un percorso guidato, mediante un wizard per intenderci, o in modalità struttura, indicando manualmente tutte le specifiche che si intende rispettare. Scegliamo la seconda strada, è più flessibile e si addice maggiormente alle caratteristiche di un programmatore. Apparirà l'ambiente visuale di figura. Selezioniamo entrambe le tabelle.

‹4› REALIZZO LA QUERY INDICANDONE I CRITERI

Supponiamo di voler conoscere i titoli delle opere (e i relativi artisti) presenti al museo Hermitage di San Pietroburgo. Cliccando sugli attributi in questione essi appariranno nella parte inferiore della finestra. Su *LUOGO* applicherò il criterio richiesto e toglierò la spunta di visualizzazione. All'uscita della query darò ad essa il nome *San Pietroburgo*.

‹5› APERTURA DELLA QUERY SAN PIETROBURGO

Terminata la fase di strutturazione della query, non rimane altro che richiamarla ed esaminare i risultati ottenuti. Nella finestra principale del nostro DB scorgeremo un nuovo arrivo, ossia la query appena costruita. San Pietroburgo è pronta per essere aperta e scoprire il risultato ottenuto, basta un colpo di clic per farlo.

‹6› ABBIAMO COSÌ OTTENUTO I RISULTATI SPERATI

SanPietroburgo : Query di selezione

NOMIN	TITOLO
Monet Claude	Prati a Giverny
Monet Claude	Il ponte di Waterloo
Gauguin Paul	Les parau parau (parole, parole)
Matisse Henri	La danza

Record: 1 di 4

Ultimo passo è quello di esaminare i risultati ottenuti. Verifichiamo che esistono delle opere presenti nel nostro database che attualmente sono tenute al museo Hermitage. Abbiamo ottenuto dalla globalità dei dati solo la porzione che ci interessava. In particolare, due opere di Monet, una di Gauguin e una di Matisse. Missione compiuta!

Come è che il desktop sarà meglio dei videogiochi

LinuxWorld 2004: Sun e Looking Glass

Nuove politiche di marketing e nuove politiche strategiche: la pubblicità si fa viva e Looking Glass diventa pubblico… Ecco un breve ma interessante aggiornamento sulle novità di casa Sun

In occasione del primo LinuxWorld italiano la Sun ha schierato un esercito di – passatemi la metafora – cartelloni umani: un'entusiasta banda di giovani vestiti di monitor e PC girava per la sede dell'evento con una demo di Looking Glass in esecuzione letteralmente sopra le loro teste, dando spiegazioni a chi, come me, incuriosito, volesse vederci più chiaro sull'intera faccenda.

Fig. 1: *La squadra di promotori del Looking Glass al LinuxWorld 2004*

Con questa iniziativa (di cui non potevamo non lasciare un ricordo: lo trovate in **Figura 1**), l'obiettivo di dare visibilità e risonanza al proprio prodotto è stato sicuramente raggiunto! Ma le sorprese non sono tutte qui: sempre nell'ottica del modello di marketing adottato, la Sun ha pensato di dare ulteriore visibilità a Looking Glass proiettandone un video promozionale su niente meno che la parete dell'edificio dell'hotel che ospitava LinuxWorld! Al di là del fatto che il buio (necessario per la proiezione) si è fatto attendere abbastanza quella sera, lo spettacolo è stato d'effetto, come potete vedere nella **Figura 2**. Comunque, è piuttosto ovvio che nessuna società metterebbe in piedi cotanta scenografia se non avesse qualcosa di importante da far arrivare al suo pubblico, ed io sono proprio qui per raccontarvi che cosa bolle in pentola in casa Sun per quanto riguarda il mondo Desktop. Avevo già accennato, in un precedente articolo dedicato alla Java Conference 2004, al progetto interno Looking Glass con cui la Sun stava lavorando alla prossima generazione di interfacce grafiche a finestre. L'idea, frutto della creatività giapponese di Hideya Kawahara, attualmente Senior Staff Engineer di Looking Glass negli Stati Uniti, è quella di portare la terza dimensione negli ambienti visuali dei sistemi operativi, partendo dall'osservazione che sono ormai 20 anni che essi non subiscono innovazioni sostanziali. Viste le sempre maggiori capacità dei processori grafici e i miracoli in termini di rappresentazione tridimensionale cui assistiamo nei videogiochi, perché non approfittare dei vantaggi offerti dalla tecnologia per migliorare il desktop di lavoro degli utenti dei PC?

Fig. 2: *Una demo di Looking Glass sulla parete del Crowne Plaza Linate*

Finestre che ruotano e che si impilano lateralmente al bordo del nostro schermo, note appiccicate sul retro delle nostre applicazioni, insomma un tangibile senso di profondità era stato annunciato dalla Sun in sede della Java Conference di giugno, seppur con tempi di attesa a medio-lungo termine. Ma al LinuxWorld ho invece scoperto che tutto questo può già essere una realtà per i più temerari!

Fig. 3: ***La home page del progetto open-source Looking Glass***

SUL WEB

Se non vedete l'ora di buttarvi nel codice e vedere cosa già c'è e cosa potete fare, ecco i link per voi: qui la guida per lo sviluppatore https://lg3d-core.dev.java.net/lg3d-developers-guide.html e qui dove scaricare tutto https://lg3d-core.dev.java.net/servlets/ProjectDocumentList?folderID=0&expandFolder=0&folderID=1849

LOOKING GLASS PER LA SUN

Ecco il sito "commerciale" della Sun relativo al progetto di desktop 3D. Scoprirete cos'è, vedrete alcuni screenshot, e c'è anche il video di una demo del prodotto allo stadio iniziale: http://wwws.sun.com/software/looking_glass/

TUTTO IL CODICE OPEN SOURCE

Looking Glass è stato messo tra i progetti open-source della comunità di *java.net*, ed è quindi scaricabile per l'utilizzo da parte di chiunque ne abbia interesse (*https://lg3d.dev.java.net/*, **Figura 3**). Convinto che ciò avrà un forte richiamo sui lettori di questa rivista, appassionati di programmazione ed innovazione, sono andato a cercare ulteriori informazioni al riguardo. Mi è stato mandato in aiuto l'Ing. Nicola Galante, della sede Sun di Padova (che colgo l'occasione per ringraziare della disponibilità), il quale mi ha illustrato caratteristiche tecniche e potenzialità di Looking Glass. Innanzitutto, si tratta di un progetto che si basa sull'architettura dei server X: esso nella sostanza si propone come un nuovo window manager in grado di creare finestre tridimensionali grazie alle API di Java 3D. La Sun concentra i suoi sforzi sul mondo Linux e Solaris e non è previsto nessun tentativo di porting su sistemi basati su Windows della Microsoft (che non utilizza il modello dei server X). Per quanto riguarda il Macintosh, invece, trattandosi di codice di sovrastruttura che si appoggia su sistemi già esistenti nel mondo Unix, è probabile che un porting sia fattibile sotto MacOS X senza ostacoli insormontabili.
Un secondo punto molto importante è che le vecchie applicazioni sviluppate per gli ambienti grafici 2D saranno perfettamente in grado di funzionare anche con il nuovo window manager: infatti grazie a modifiche del protocollo X – che è alla base della comunicazione tra lo X Server e le applicazioni che visualizzano elementi sul desktop – sarà possibile captare le richieste grafiche dei programmi bidimensionali e creare delle finestre 3D per visualizzarne il contenuto. Chiaramente le nuove applicazioni non saranno in grado di avvantaggiarsi di tutte le caratteristiche che Looking Glass mette a disposizione, ma alcune, quali le note dell'utente sul retro delle finestre, saranno comunque utilizzabili: tali note infatti sono memorizzate nel file system in base al processo a cui vengono associate, indipendentemente dal tipo di programma.
Chiaramente c'è molto entusiasmo per questo progetto nella comunità open-source. Al di là del filone principale (*https://lg3d-core.dev .java.net/* e *https://lg3d-x11.dev.java.net/*), che si occupa del motore 3D e dell'integrazione con il server X, sono già partiti una serie di sottoprogetti paralleli per creare le classiche applicazioni di qualunque desktop (music player, notepad, calcolatrice, etc - *https://lg3d-demo-apps.dev.java.net/*) oltre allo spazio dedicato alla grafica e alle animazioni (*https://lg3d-art.dev.java.net/*). Infine, in parallelo a tutti i progetti rilasciati su *java.net*, la Sun ha avviato internamente, seppur in fase ancora decisamente embrionale, i lavori di revisione di tutta la libreria Swing per renderla compatibile con il nuovo desktop tridimensionale: questo avrà un enorme impatto per gli sviluppatori Java, che vedranno le loro applicazioni Swing trasformarsi magicamente in applicazioni a tre dimensioni con nessun o poco sforzo semplicemente passando da un desktop standard al nuovo Looking Glass. Se vogliamo sognare un po' in avanti, la nuova tecnologia aprirà sicuramente nuove frontiere anche dal punto di vista dell'hardware: per esempio, potranno conquistare mercato i mouse 3D, già esistenti ma di uso limitato ad ambiti molto specializzati, quali il design aeronautico. È difficile immaginare cosa potrà sedere sulla nostra scrivania nei prossimi anni…
Tirando le somme di tante novità vorrei concludere facendo notare come quella che a giugno era solo una promessa sia oggi divenuta una tangibile realtà grazie alla politica della Sun di offrire il progetto Looking Glass alla community open-source. L'invito a questo punto è d'obbligo: tenete d'occhio gli sviluppi del nuovo desktop e soprattutto… partecipate!
Dimostrate che il nostro paese è ancora una culla di creatività e talento dando un apporto significativo a quello che sicuramente sarà il futuro dei desktop dei personal computer!

Federico Mestrone

Biblioteca

ON LINE

WEBERDEV

Un sito dedicato a tutti gli sviluppatori PHP e MySQL. Una grande raccolta di risorse con particolare riferimento a codice ed articoli. Aggiornato frequentemente, permette di rimanere in contatto con tutte le novità della piattaforma.

www.weberdev.com/

JAVA WORLD

News,tips, libri, articoli e tutto quan'altro possa essere di interesse per i programmatori Java. Ospita una collezione di articoli fin dal lontano 1986! Un punto di riferimento, che si è conquistato la fiducia dei lettori con articoli molto approfonditi e semplici da seguire.

www.javaworld.com/

ABSTRACT VB

Articoli, codice, tutorial, recensioni di libri e download per il linguaggio VB e per la nuova piattaforma Microsoft .NET

http://abstractvb.com/

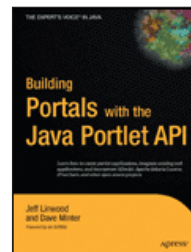
C – CORSO COMPLETO DI PROGRAMMAZIONE II EDIZIONE

Un testo che, già nella prima edizione, è diventato un vero e proprio standard di riferimento per l'insegnamento della programmazione in linguaggio C. Il testo introduce, con gradualità, ai concetti fondamentali della programmazione (strutture di controllo, funzioni, array, puntatori, gestione dell'input/ output, strutture dati), offrendo al tempo stesso una panoramica ampia e articolata del linguaggio C standard.
In questa seconda edizione la presentazione dei concetti e degli esempi svolti è stata ulteriormente affinata, rendendo il testo ancora più chiaro ed efficace. Trai diversi contenuti: introduzione alla programmazione in C, sviluppo di programmi strutturati, il controllo dei programmi, le funzioni, i vettori, i puntatori, i caratteri e le stringhe, la formattazione dell'input/output, le strutture,ecc.

Difficoltà: Medio-Alta • Autori: H.M. Deite – P.J.Deitel • Editore: Apogeo
www.apogeonline.com ISBN: 88-503-2254-2 • Anno di pubblicazione: 2004 • Lingua: Italiano
Pagine: 556
Prezzo: € 35,00

BUILDING PORTALS WITH THE JAVA PORTLET API

Come si sviluppano applicazioni per portali? Come si integrano all'interno di un portale sistemi di amministrazione e motori di ricerca? Jeff Linwood e Dave Minter in questo testo mostrano come risolvere questi problemi utilizzando le Java Portlet API.
Nel testo vengono approfondite tutte le problematiche riguardanti la sicurezza e l'interazione delle portlet con le servlet e API, esplorando concetti e codice con esempi pratici.

Difficoltà: Alta • Autori: J.Linwood – D.Minter • Editore: Apress www.apress.com
ISBN: 1-59059-284-0 • Anno di pubblicazione: 2004 • Lingua: Inglese • Pagine: 393
Prezzo: $ 49.99

APACHE COOKBOOK

Apache Cookbook è un testo che si pone come guida con soluzioni ed esempi pratici per i webmaster, gli amministratori web, i programmatori e in generale per chi lavori con il Web Server Apache. Ad ogni problema viene contrapposta una soluzione sperimentata praticamente, corredata da codice e facilmente riutilizzabile. Gli argomenti trattati spaziano dall'installazione del server, a questioni più complesse, come la gestione e la messa in sicurezza di un server proxy, la scelta di configurazioni performanti e le tecniche di protezione tramite password. In aggiunta, il testo contiene documentazione per:

- redirezionare e riscrivere URL;
- negare l'accesso a richieste non referenziate;
- eseguire script CGI come se si fosse proprietari del file;
- determinare la quantità di memoria necessaria al proprio server;
- ottimizzare i link simbolici e la creazione dei processi.

Difficoltà: Alta • Autori: K.Coar – R.Bowen • Editore: Hops Libri www.hopslibri.com
ISBN: 88-481-1652-3 • Anno di pubblicazione: 2004 • Lingua: Italiano • Pagine: 238
Prezzo: € 29,90